national center for women & INFORMATION TECHNOLOGY™

PAIR PROGRAMMING-IN-A-BOX

The Power of Collaborative Learning

BREN:ICS INFORMATION AND COMPUTER SCIENCES UCIRVINE UNIVERSITY of CALIFORNIA

Pair Programming-in-a-Box helps you improve college students' programming competency. Designed for easy customization and implementation, this Box provides tested methods for implementing pair programming, a technique proven to increase the performance and retention of computing students. The materials in this box help faculty introduce the program to students and staff, form programming pairs, assess students' individual competency, and evaluate effectiveness.

NATIONAL SCIENCE FOUNDATION

AVAYA

Bank of America

Microsoft

Pfizer

Pair Programming-at-a-Glance

WHAT IS PAIR PROGRAMMING?

Pair programming is a collaborative learning method in which students program in pairs instead of individually. This approach significantly improves college students' programming competency and increases the likelihood that both male and female students become and remain computer science majors.

During pair programming, students work in tandem at one computer while completing regular programming assignments. The "driver" controls the mouse and keyboard while the "navigator" makes suggestions, points out errors, and asks questions. The partners routinely switch roles to gain the benefits of each role.

"My partner had never coded anything before so I was able to teach him a little bit about how it worked. The teaching bit helped me a lot with understanding the labs and passing the exams." ~ CS Student

WHY PAIR PROGRAMMING?

Improves Computer Science (CS) Enrollment, Retention, and Students' Performance

- » Research results show that pair programming increases the percentage of introductory students who declare a CS major.
- » Research shows that pair programming increases the number of students who remain in the CS major one year later, as compared to their non-paired peers.
- » Pairs of programmers often produce higher-quality work in less time than when programmers work individually, even if one partner is stronger than the other.
- » Students have an opportunity to read another programmer's code, uncovering flaws in their work and deepening their understanding.

Increases Confidence

- » Research results show that pair programming reduces the so-called "confidence gap" between female and male students, while increasing the programming confidence of all students.
- » Pair programming supports the natural development of supportive peer groups, a critical structure for helping students succeed in current and subsequent courses and on into the workplace.

Develops Critical Skills

- » Critical thinking
- » Social aptitude
- » Professional capacity
- » Teamwork
- » Communication proficiency

Minimal Effort Required by Course Staff

- » There is no need to modify regular course assignments; students will complete assignments familiar to course staff.
- » Student pairs are likely to think through and resolve difficulties before coming to the instructor or lab assistant, possibly easing the workload of course staff.

Introduces Students to the "Real World" Working Environment

The image of the solitary programmer is a thing of the past. Companies increasingly expect employees to work in collaborative teams. In fact, pair programming is being employed more frequently in industry because it's known to improve the software development process.

Including collaborative learning experiences such as pair programming from the very start of CS undergraduate students' academic career, exposes them to a valuable "real world" environment.

ARE YOU READY TO INTEGRATE PAIR PROGRAMMING INTO YOUR CS COURSE?

Use materials in *Pair Programming-in-a-Box: The Power of Collaborative Learning* to get started, and to support staff and students each step of the way.

national center for women & INFORMATION TECHNOLOGY

PAIR PROGRAMMING-IN-A-BOX

The Power of Collaborative Learning

WHAT IS *PAIR PROGRAMMINGING-IN-A-BOX?*

Pair Programming-in-a-Box **is a set of resources that helps instructors of introductory college programming classes to easily integrate pair programming into their courses. The "Box" provides tested, validated methods for implementing pair programming, including orienting students and course staff, forming programming pairs, facilitating teamwork, assessing students' individual competency, and evaluating the effectiveness of pair programming.**

INSIDE THE BOX

Pair Programming-in-a-Box provides all the components you need to successfully use pair programming methods in your courses. These materials can be readily adapted to meet the particular needs of a course:

- an overview of the principles, benefits, and activities of pair programming
- a guide for integrating pair programming into the course
- a sample handout for students that explains pair programming, describes how pair programming activities are factored into their course grade, and guides students on such topics as choosing partners and dealing with interpersonal differences
- a guide for instructional staff that prepares them for their role in a course that uses pair programming
- a comprehensive survey to gather data on students' pair programming experiences and opinions, suitable for performing data analysis, including analysis along demographic dimensions
- a resource guide that summarizes the research on pair programming and suggests supplementary course materials, videos, and readings

RESOURCES AT THE READY

Pair Programming-in-a-Box: The Power of Collaborative Learning is available at **www.ncwit.org/pairprogramming.** Download the entire "Box" as a single, compressed folder, or select individual components to use as you need them.

Pair Programming-in-a-Box *was developed by NCWIT in collaboration with the Ada Byron Research Center, Donald Bren School of Information and Computer Sciences, University of California–Irvine.*

Pair Programming-in-a-Box *was created by Norman Jacobson, University of California, Irvine and edited by Jane Krauss, NCWIT. Thanks to the following people for their significant contributions: David Kay, University of California, Irvine; Laurie Williams, North Carolina State University; Vicki Almstrum, University of Texas at Austin; Suzanne Schaefer, University of California, Irvine; Charlie McDowell, University of California, Santa Cruz; Lecia Barker, University of Texas at Austin; and Joanne Cohoon, University of Virginia.*

Table of Contents

Pair Programming At-a-Glance **2**

Overview of Pair Programming in Introductory Programming Classes **5**
- Pair programming basics 5
- Pair programming's many benefits 5
- Localized implementation 6
- Forming pairs 6
- Minimal adjustments to the course reap significant benefits 6
- Getting the most out of pair programming 6

Integrating Pair Programming into the Course **8**
- Lab sections 8
- Preparing students for pair programming 8
- Assessing students' programming competency 9
- Pairing students 9
- Dealing with a faltering partnership 9
- Short-duration pairing and a regular schedule 10
- Forming new partnerships and routine switching of partners 10
- Coordinating pair formation with drop deadlines, due dates, and exams 11
- Using the pair programming questionnaire and survey 12
- Awarding points for pair programming activities 12

Preparing Course Staff for Pair Programming **13**

Sample Handouts for an Introductory Course **14**

Pair Programming in CS1 — Sample Student Handout **15**

Pair Programming Guidelines — Course Staff Handout **17**

Sample Pair Programming Questionnaire **20**

Sample Student Survey **23**

Resources and Acknowledgments **28**
- Resources 28
- Acknowledgments 31

Evaluating *NCWIT Pair Programming-in-a-Box: The Power of Collaborative Learning* **32**

Overview of Pair Programming in Introductory Programming Classes

This overview describes the basic principles, benefits, and activities of pair programming. Throughout this program-in-a-box, a bracketed number references an information source found in the Resources Guide.

PAIR PROGRAMMING BASICS

Pair programming in the classroom involves having students pair up at a single computer to complete programming work, using these pair programming methods:

- One member of the pair, the "driver," performs the "on-computer" tasks needed to complete the program, including controlling the mouse and keyboard to enter, compile, and run the code.
- The other member of the pair, the "navigator," actively watches the driver's activities, makes suggestions, points out errors and problems, asks questions, and thinks about slightly longer-term strategies.
- The driver typically responds to the navigator's input by taking actions at the computer or engaging the navigator in further conversation that moves the project forward.
- Every 20 minutes or so, the partners switch roles, ensuring students work as a true team and obtain the benefit of acting in each role. The pair continues working in this manner until the programming work is complete.

Following the pair programming model is not effortless for students, yet benefits depend on following its methods closely. One common (but not the only) approach that ensures students follow the model is for the course to include required and regularly scheduled "lab sections," work times when the pairs collaborate in a computer lab with a course staff member present to monitor and assist them. Please note: Critical methods presented in this resource, such as pairing students and assessing individual progress, are predicated on the assumption that required and regular lab sections are in place.

PAIR PROGRAMMING'S MANY BENEFITS

Extensive study of pair programming in the classroom over several years has provided compelling evidence that pair programming significantly benefits students. Some research indicates that novice students gain the most from pair programming, yet advanced students benefit significantly, too [2].

Compared to "solo" programming, pair programming:

- Improves the quality of students' programs
- Produces consistently higher-quality work, in less time, than when programmers work individually, even if one partner is stronger than the other
- Builds on partners' strengths by allowing them opportunities to solidify and deepen their own understanding; it frequently uncovers flaws in the work
- Helps overcome weaknesses by providing a peer who can immediately address questions and difficulties
- Provides students a direct and immediate opportunity to read another programmer's code, an important programming skill not possible when programming alone
- Increases students' confidence with programming
- Tends to engage students more actively and directly in the course
- Increases the number of students who complete the course
- Increases the percentage of introductory students who declare a computer science major
- When used in CS1, leads to increased solo programming competency in CS2
- Is more enjoyable than programming alone for most students, increasing their satisfaction with the course
- Supports the natural development of supportive peer groups, a critical structure for helping students succeed in current and subsequent courses and on into the workplace
- Gives students early collaborative work experience, fostering the development of teamwork and communication skills that are highly valued in both the academic realm and the job market
- Encourages students to turn to their partners as the first option in obtaining help with their programming; this rapid assistance improves productivity, increases a sense of accomplishment, and reduces reliance on course staff

LOCALIZED IMPLEMENTATION

Other activities can be included along with the basic ones students undertake:

- The same pair of students can work on all assignments, or a new pair can be formed for each assignment or group of assignments.
- Each student can receive the same or an adjusted grade for work the pair completes, or, to assure that each student's programming competency is individually assessed, the work completed by the pair can be ungraded, serving as preparation for a graded assignment or lab exam students take individually. Read more about assessing students' programming competency in *Integrating Pair Programming into the Course*.
- Students can be asked or required to complete a short pair programming questionnaire to review their own and their partner's performance and to assess the pair programming experience overall. This information can be useful for an instructor who is factoring students' pair programming performance into their grades and for improving the implementation of pair programming in the course. *See Sample Pair Programming Questionnaire*. Students also can complete a more comprehensive survey about these and related matters, the results of which are suitable for formal data analysis. See the *Sample Student Survey.*
- An instructor can give or take away points based on students' participation in pair programming and/or for completing the pair programming questionnaire and survey. Guidance for factoring these elements into the grade is described in *Integrating Pair Programming into the Course*.

FORMING PAIRS

Pairs must function well for students to gain the most from pair programming. The following pair formation methods result in a very high percentage of compatible pairings:

- Form pairs based on students' common levels of programming experience.
- Pair students randomly.
- Allow students, with guidance, to form their own pairs.

See *Integrating Pair Programming into the Course* for more information about the advantages and disadvantages of these pairing methods.

MINIMAL ADJUSTMENTS TO THE COURSE REAP SIGNIFICANT BENEFITS

Obtaining the significant benefits of pair programming requires thoughtful integration of these methods and some updating of course materials; these tasks take a minimal amount of time and effort.

Most important to note is that the programming assignments for the course need not change. Students just work in pairs instead of individually to complete the work; the content of assignments is not affected.

To remove any doubt about how well a grade awarded for paired work reflects the competencies of individuals, the instructor may choose to develop assignments or tests that students complete individually [5, 24, 25].

Preparing students to undertake pair programming requires no special training or practice; once students know what they are supposed to do and have a partner, they can quickly and easily begin. See *Pair Programming in CS1 — Sample Student Handout* to see how the process and expectations can be explained to the class.

Preparing teaching assistants and lab tutors to work with students takes no specific training either. Integrating discussion of pair programming into regular orientation talks and training is sufficient. *Preparing Course Staff for Pair Programming* describes the duties course staff typically undertakes, and offers ways to ensure staff is adequately prepared. *Pair Programming Guidelines — Course Staff Handout* spells out their responsibilities.

GETTING THE MOST OUT OF PAIR PROGRAMMING

To achieve the benefits of pair programming, instructors should:

- Make sure students follow the pair programming model
- Strive to improve the performance of weak partnerships
- Ensure individuals' programming competency is accurately assessed

- » Encourage students to complete pair programming questionnaires and surveys
- » Make sure feedback on the surveys is fair and complete

The component *Integrating Pair Programming into the Course* discusses pair programming and its variations, benefits, and drawbacks, and offers observations and recommendations to help an instructor plan for and implement pair programming. Resources [5] through [23], and the references given in resources [30] to [32], discuss aspects of these topics even more deeply.

NOTES:

Integrating Pair Programming into the Course

Pair programming can be successfully implemented in several ways. The approaches described here have worked well. Use this guide as you plan to integrate pair programming into your introductory computer science course.

LAB SECTIONS

Pair programming is not intuitive to students; most have been deeply socialized to think that working alone or in an *ad hoc* partnership is sufficient for doing good work. Following the model takes a new kind of effort that working alone or in informal partnerships does not require, so students need reminders to work with their partner seriously, take on the roles of driver and navigator faithfully, and to switch roles frequently (every twenty minutes or so). While this requires more cognitive effort, students should be reminded that the payoff in terms of developing in-demand professional skills is substantial.

The pair programming model is straightforward to implement in a course that has lab sections with these characteristics:

- Lab sessions meet regularly and frequently, so pairs can more easily coordinate their schedules.
- Lab sessions meet in a room where students can easily pair up at computers.
- Lab attendance is required. (You can also provide optional lab sections, but it is important that attendance is required at some sessions.)
- At least one course staff member is present to monitor and assist pairs (e.g., a teaching assistant).

This arrangement assures students put in the time needed to learn to program well and that their adherence to the pair programming model is monitored and supported. The staff member can assist students with their work, intervene if a pair is not following the model, and remind students when it is time to switch roles. A reminder may be needed because students sometimes get wrapped up in their work and forget to switch or occasionally prefer to stick with what they are good at. Yet, playing to their strengths denies students the benefits gained by taking on the other role.

If students are evaluated on their performance as programming partners, lab sections provide the best opportunity to watch the pairs in action and to learn how well or poorly partners are adhering to the pair programming model.

Of course, many students will work on their own, away from lab, if only to ponder a difficult aspect of the assignment. Such self-study is useful, but students should be encouraged to call a pair meeting to tackle difficult problems using the pair programming approach. Quick resolution of sticky problems through pair programming helps students see the advantage of collaboration over working alone. If a meeting is not feasible, students should be encouraged to discuss the problem with their partner at the first opportunity.

Recent research indicates that students can obtain the benefits of pair programming when physically separated if suitable collaboration software is available to support the pair programming model [9]. Such distributed pair programming in a course is a relatively new approach and not yet widely tested; if used instead of lab sections, it is likely more difficult to ensure that students actually program in pairs. Still, instituting distributed pair programming along with lab sections could well encourage students to work in pairs when away from the lab.

PREPARING STUDENTS FOR PAIR PROGRAMMING

Students require no special training or exercises to undertake pair programming. Though some have suggested that a tutorial helps students to obtain the benefits of pair programming (see the appendix of [10] for an example), several others have noted that students as a whole undertake pair programming without complaint or difficulty as long as the benefits and requirements of pair programming are well explained and students are clearly aware that pair programming is a course requirement [5].

To aid the introduction of pair programming and to show pairs in action, an instructor might show this accurate (and entertaining) introduction to pair programming from North Carolina State University: *An Introduction to Pair Programming for Students* [33].

It is best to address pair programming in course materials, particularly in the course syllabus and programming assignments, and to describe pair programming and the particulars of its implementation at the first lecture or class meeting. It is important to remind students throughout the course why pair programming

is worth doing and to promptly address any issues or problems that arise. The instructor should let students know that course staff will keep the instructor apprised of lab activity so that common issues with the class or individual concerns can be addressed as warranted.

The sample handout in this resource called *Pair Programming in CS1 — Sample Student Handout* has the information course staff might impart when introducing pair programming. Other examples from actual courses are found in [37-42].

It is worthwhile to go over the peer survey with students at the time pair programming is introduced. Since the survey includes questions that embody the qualities and behaviors of successful pair programming, its discussion reinforces to students what actions make for good pair programming. Also, if points can be earned for successful pair programming, this is a fine time to discuss the rubric for awarding them.

ASSESSING STUDENTS' PROGRAMMING COMPETENCY

Perhaps the strongest objection made against the adoption of pair programming by experienced instructors is the concern about being unable to correctly gauge the programming competency of individuals. The basis of this statement is generally that one student in the pair could be much more experienced at programming and has either "covered" for the other or dominated, so that the less experienced student doesn't get a chance to learn. In this case, making a joint grade for both members would be inaccurate and unfair. However, one can also argue that if the pair programming model is followed, both students had equal input into the quality and correctness of the program, so the grade is fairly applied to each. Under this model, it becomes more important to guide and monitor the pair programming process.

Pair programming and the assessment of each individual's programming competency need not conflict. The easiest solution is to allow students to complete ungraded assignments in pairs and then assign minimally different assignments for individuals to complete for a grade. Another suitable approach is to have individuals take a graded exam in which they code some or all of a program that is similar to the assignment they just completed jointly. Coding exams completed on the computer are preferred, but paper-based exams can suffice, too.

Several minimally different approaches have been used [24, 25] but all are designed so that programming pairs are not graded jointly on assignments.

PAIRING STUDENTS

A significant concern in pair programming is how to form compatible pairs. Well-functioning pairs gain the benefits of pair programming and simply enjoy the course more than students who have a problematic relationship with their partners. Certainly, having a large number of malfunctioning pairs reduces the degree of success in using pair programming in a course.

Recent research and practice offer three approaches that produce a high rate of compatible partnerships. Different pairing methods may suit different course configurations or composition:

1. *Students are paired randomly.* In a study at North Carolina State University, researchers found that 93% of pairs report that they worked compatibly with their partners when randomly paired [23]. Other research [e.g., 13] also suggests that a significant majority of partnerships will be compatible without any proactive attempt to make them so.
2. *Students are paired based on similar work ethic, compatible learning style, or perceptions of each other as having a similar skill level* [23]. Using these criteria to form the pairs should result in a very high percentage of compatible pairings, but will likely require a significant time commitment for a large-enrollment course.
3. *Students choose their own partners*. This method works best when students are given guidance in choosing a compatible partner and given some days to get to know one another before they are officially assigned as partners. For the rare student who cannot find a partner, the instructor or assistant pairs him or her. This approach has the added benefit of giving students opportunities to get to know one another, often resulting in the development of peer support groups shown by research to help students succeed in computer science majors [5].

DEALING WITH A FALTERING PARTNERSHIP

"I love working with other people, but not my specific partner." – CS1 student

Sometimes students enter into a partnership that doesn't go smoothly. Partners may find they are incompatible, or one person stops participating. In these instances, course staff needs to intervene to improve the situation. Fortunately, instances of faltering relationships are rare, so the time needed to deal with problem pairings is minimal.

Approaches to dealing with problem partnerships include:

» Make it clear in course materials and discussions that problems within partnerships sometimes occur and that it is normal for people working together for the first time to have issues. Describe the most common problems partners face and recommend ways pairs might go about resolving them. If partners cannot solve a problem on their own, advise students to come to staff for guidance. Describe the procedure to be used for dealing with pairs who have issues with one another.

» If a student or pair reports a problem, advise the partners to try to work things out, and give guidance on how to do this effectively and courteously.

» If partners are not able to discuss the problem or if their discussions fail to solve the problem, intervene directly to repair the partnership.

» If staff intervention fails to resolve the issue with a reasonable amount of time and effort, permit a "divorce," where each student is allowed to work alone or with another pair (more on this below) until each student can be paired with a more compatible partner.

SHORT-DURATION PAIRING AND A REGULAR SCHEDULE

One advantage of assigning partnerships of short duration (for example, changing partners with every weekly assignment) is that faltering partnerships don't last long. Knowing they will have a new partner in a few days can give students incentive to make things work for the short term.

A significant source of conflict in a partnership is the inability for partners to find time to meet with each other. Requiring student attendance at the frequent, regularly scheduled lab sections (recommended above) can minimize this problem.

FORMING NEW PARTNERSHIPS AND ROUTINE SWITCHING OF PARTNERS

Inevitably, some partnerships end prematurely. The most common cause is when one member drops the course or stops participating (without officially dropping the course). The other instance is when course staff allows a pair to find new partners.

Unpaired students need to be placed into new compatible pairs as soon as is feasible. Here are some tested options:

» Students are allowed to work solo until the class is scheduled to form new partnerships (see below). If new partnerships are formed frequently, the student is not working alone for long.

» If two or more pairs have dissolved, new pairings are made among these individuals.

» A student is made the third member of a willing team until a new pair can be formed. The triad operates with one driver and two navigators and all cycle through the driver role. This arrangement can become cumbersome, so it is best kept only for a short time.

Another issue to address is how often students should rotate partners. In courses where students complete one large programming project, an obvious and sensible approach is for students to remain in the same pairing for the duration of the project.

In courses with multiple assignments, have students switch partners after every assignment, or when a group of two or three related assignments is completed and the next unrelated set begins. Each approach has its advantages and drawbacks:

» Not switching at all allows students to develop a deeper working relationship with their partner and gives them more opportunity to work through problems that may arise, which is a useful interpersonal skill. Not switching also keeps to a minimum the time course staff devotes to forming and recording pairings.

» Switching partners at least once roughly models what occurs in the workplace, where programmers work on a team for a while then find themselves working with new additions or on a new team entirely. Switching partners also places solo students back into pairs, and importantly, provides a "way out" for a student in a less-than-optimal pairing who chooses not to bring the matter to staff's attention. Switching pairs has another benefit, too. It gives the teaching staff multiple views of the contributions of each student. If several students complain about the same student, the course staff knows the student is a problem – and it's not one person's word against the other.

» Switching too often does not allow a student to settle into a good working relationship with a partner; the overhead to the student in adapting to a new relationship can reduce the productive time the pair has to work on assignments. Additionally, the overhead of recording and evaluating each student's performance in numerous pairings could strain course resources.

If assigning closely related assignments, it is best to wait to switch partners until that set of assignments is complete. Changing partners each assignment within a group of closely related assignments forces a new pair to rebuild the program their previous partnership had already constructed, diverting attention and time from completing the new assignment.

On balance, in courses where there are multiple assignments, switching at least once is preferable to not switching all.

COORDINATING PAIR FORMATION WITH DROP DEADLINES, DUE DATES, AND EXAMS

It is a good idea to coordinate pair formation with deadlines for dropping classes, assignment due dates, and class exams to reduce the time unpaired students work alone or as a third team member. Timing assignments to course drop deadlines also makes pair assignments more coherent.

To illustrate, consider the example below [adapted from 5]:

> Consider a ten-week introductory programming course where students complete six programming assignments and in which students are allowed to drop through the end of the sixth week.

Imagine the first large group of drops occurring in the first two weeks of the course as students decide if the class is appropriate for them. While students may work in *ad hoc* partnerships during this time, the deadline for official partner formation is set to the end of week two and the first assignment is due in week three. (This may seem late in the course, but it does take some time for students to get ready for the learning that is to come. Orientation exercises that allow students to get familiar with the programming environment and the basics of editing, compiling, and running programs are helpful during this time.)

In this scenario, if a student drops during the first week, no official pair is broken. If a student drops during the second week, the remaining student has had some pair experience, has completed some work on the assignment, and has a chance to pair up with another student who also needs a partner; they can share their work to date, which helps the new partner complete the assignment on time.

Suppose the last day to drop the course occurs at the end of week six. In this case, one would set assignment due dates at the middle of week seven and also form new partnerships during this week, after the completion of the current assignment.

A student losing a partner during week six has still done about a week's worth of pair work on the assignment. The solo student may pair up with another student who has lost a partner, or may work alone for the few days before all students form new pairs in the middle of week seven.

INTRODUCTION TO PROGRAMMING									
	Form Partners →	Assignment Due			Drop Deadline →	Assignment Due → Form new partners			
Week 1	Week 2	Week 3	Week 4	Week 5	Week 6	Week 7	Week 8	Week 9	Week 10

USING THE PAIR PROGRAMMING QUESTIONNAIRE AND SURVEY

Instructors report that surveying students about their partnerships and getting their impressions about their pair programming experience is helpful in determining appropriate credit for students' pair programming participation and for improving the way pair programming is implemented in the course. Surveying students is particularly important when they are expected to work with their partners outside of class. Understanding that paired work will be monitored in this way is likely to motivate students to contribute equitably.

The *Sample Pair Programming Questionnaire* is a short instrument students can quickly complete by filling in a web form or composing an email message. The questionnaire asks about the partner's as well as the individual student's participation, and assesses impressions of the pair programming method.

The *Sample Student Survey* is a more extensive instrument that includes several in-depth questions about pair programming, partners, and course staff; it also collects demographic data. The survey questions allow for a more extensive analysis than possible with the more limited questionnaire.

These instruments allow students to provide specific feedback about pair programming implementation in the course and any particular issues that might have occurred with their partnership. The information is helpful in identifying and resolving problems relating to implementation of pair programming in the current course and improving the implementation of pair programming in subsequent course offerings.

AWARDING POINTS FOR PAIR PROGRAMMING ACTIVITIES

Awarding even just a few points for pair programming serves as a strong motivator for most students. If points are deducted for avoiding paired work or for participating poorly, students understand that pair programming is an important aspect of the course and are more likely to do it well. (Some instructors use a multiplier instead of points. If a student only does half the work, he or she receives half the grade.) Finally, if points are awarded for completing evaluations and the end-of-term survey, more students will complete them. Some instructors have different views about assigning points in this way, but this system has worked within the model of pair programming presented here.

NOTES:

Preparing Course Staff for Pair Programming

In a course where teaching assistants or lab tutors will oversee pair programming, the instructor must prepare the staff for their roles and responsibilities.

Teaching assistants and lab tutors who have engaged in pair programming as students bring welcome experience to the course. Yet, perhaps surprisingly, neither previous experience with pair programming nor specialized training is necessary to ensure that staff members perform their pair programming duties well. However, it is crucial that they are informed of their duties in detail and they understand the value added by pair programming.

When familiarizing staff with their duties it is helpful to provide a written description. For an example, see *Pair Programming Guidelines — Course Staff Handout*. Instructors or teaching assistants supporting programming pairs will typically perform these tasks:

» Assign pairs, if pairs are to be randomly selected, or assist students in finding a partner if they are forming their own pairs.

» Explain the processes for pair programming and distribute *Pair Programming in CS1 — Sample Student Handout*.

» Answer questions about the programming assignments, programming language, and environment.

» Monitor pairs to ensure they are following the pair programming model.

» Note any pairs that have difficulty working together; as appropriate, either intervene or report the problem to the instructor.

» Review completed pair programming questionnaires and record any points given for their completion.

» In consultation with the lab tutors (if the course has them), report to the instructor which pairs have done particularly well or poorly in following the pair programming model.

Lab tutors, if available to a course, typically perform fewer tasks than an instructor or teaching assistant. Their tasks related to pair programming may be limited to these tasks:

» Answer questions about the programming assignments, programming language, and programming environment.

» Monitor the pairs, noting which are performing well or poorly, and share this information with the course staff.

Sample Handouts for an Introductory Course

The sample handouts on the next pages can be modified for use with students and course staff. These handouts are based on a standard introduction to computer science course (often called CS1) with the following characteristics:

- The course staff consists of an instructor and sometimes one or more teaching assistants or lab tutors. If the instructor has no support staff, he or she takes on the duties described.
- The course has required, scheduled labs during which student activity is monitored by staff.
- Students choose their own partners with guidance from course materials and staff. (Sample handouts are modified if pairs are assigned.)
- There are six programming assignments; the first three are closely related to each other, as are the last three. On balance, this scenario provides advantages compared to other approaches (see *Integrating Pair Programming into the Course* for details).
- Students are awarded a few points towards their course grade for completing the student pair programming questionnaire and docked a few points if they are poor partners or do not follow the pair programming model. As discussed in *Integrating Pair Programming into the Course,* awarding or deducting points motivates students to follow the model and complete the questionnaire.
- The six assignments are not graded; rather, students individually complete graded exams where they code all or part of each assignment. This approach avoids the concern that a grade for joint work might not correctly reflect the capabilities of an individual. (Again, see *Integrating Pair Programming into the Course* for details.)

Pair Programming in CS1 — Sample Student Handout

[adapted from 38-39]

An explanation to students of pair programming, how to choose a partner, how to deal with potential differences, and how pair programming activities are factored into the course grade. If you are viewing this document in pdf format, be aware that this file is also available as a modifiable Microsoft Word (.doc) at ***www.ncwit.org/pairprogramming****. Modify this information to suit the course and distribute it as a handout or as part of the course syllabus.*

PAIR PROGRAMMING BASICS

When working on programming assignments, whether in lab or elsewhere, you will use the "Pair Programming" approach. In pair programming, two programmers share one computer. One student is the "driver," who controls the keyboard and mouse. The other is the "navigator," who observes, asks questions, suggests solutions, and thinks about slightly longer-term strategies. The two programmers switch roles about every 20 minutes. Working in pairs should make you much better at programming than would working alone. The resulting work of pair programming nearly always outshines that of the solitary programmer, with pairs producing better code in less time.

> *"[Pair programming] makes learning programming faster and more fun. I have had previous experience working both alone and with partners. I definitely agree that working with partners is more profitable." – CS student*

To "learn the do's and don'ts" of pair programming and to see pairs in action, view this entertaining video about pair programming from North Carolina State University: *An Introduction to Pair Programming for Students* [33].

You will work with one partner for the first three assignments and a different partner for the last three. For each of these two pairings, we expect students to choose their partners. If you prefer not to choose your own partner, or cannot find a partner, let your teaching assistant know; she or he will find a partner for you. You and your partner must report the partnership to your teaching assistant no later than the dates given in the Course Schedule.

You will likely need to arrange times to meet outside of class. We expect everyone to be flexible and professional in arranging those times as necessary; if your schedule is highly constrained, explore possible meeting times with your prospective partner before you commit to the partnership.

CHOOSING YOUR FIRST PARTNER

Your partner must be assigned to your same lab section, unless you get explicit expressed permission from the instructor to partner with someone in another section. This will only be allowed under unusual circumstances, such as when there is an odd number of people in your section and all others are paired up.

If it is absolutely infeasible for you to work with a partner, see the instructor. If the instructor determines that there is no other option, she or he may give you permission to work alone. This is rarely allowed. If applicable, you must report your solo status to course staff by the same date that pairs report their partnership.

You should try to pick a partner whose experience and skill level with programming is similar to your own. This may not always be possible and it is sometimes hard to compare skill levels, but students tell us (and other data support) that pairs are most productive when the partners start at about the same level.

Still, people often pair up with another whose skills are different. This happens more often than not, as no two people have an identical skill set. The differences may be great or small, but this is exactly like most real-world working situations. Part of accomplishing a task is to get the most out of each member and make each member stronger and more productive on subsequent tasks.

Students bring different strengths to the process, regardless of how much experience they have had with programming. Both experienced and inexperienced students will need to draw on their reasoning and problem solving skills. A more experienced partner may sometimes feel frustrated or slowed down by a less experienced partner, but the experienced partner still benefits from the teamwork in many ways. The less experienced partner's requests for clarification often uncover flaws in an approach or solution; the exercise of providing a clear explanation solidifies

and deepens the explainer's own understanding and the teamwork and communication skills they gain have great value in both the academic realm and the job market.

> *"My partner had never coded anything before so I was able to teach him a little bit about how it worked. The teaching bit helped me a lot with understanding the labs and passing the exams."*
> *– CS student*

The less experienced partner may feel that questions hold the other partner back or that there is no benefit to participating actively, but pair programming studies show that paired work is consistently better than work the stronger partner does alone. It is each partner's job to understand the whole task; that means asking questions when necessary and answering them when possible.

It may be instructive to read a selection of students' partner evaluation comments from similar courses that have used pair programming; they give a picture of what good partnerships are like (and a few disasters, too). Read *Pair Programming Evaluations* from the University of California, Irvine Bren School of Information and Computer Sciences [42].

DEALING WITH DIFFERENCES

If you believe your partner is not participating appropriately in pair programming (e.g., she or he often does not come to lab, does not keep in touch, or does not come to lab prepared to work) please first address your concerns to your partner, and try to agree on what should be done to make the pair programming experience work well for both of you. If that approach is not successful, explain the issues to your teaching assistant or the instructor, who will work with you and your partner to improve the situation.

If your partner drops the course or otherwise stops participating, notify your instructor or teaching assistant immediately so he or she can pair you with another student. In rare cases you may be asked to join another partnership or, as a last resort, to work alone until a suitable partner can be found.

HOW PAIR PROGRAMMING AFFECTS YOUR GRADE

All graded items in this course are completed individually, so how your partner scores on these items will not affect your score on them. That noted, lab attendance and completion of lab assignments is critical to your test performance, so you are expected to work collaboratively with a partner using the pair programming methods outlined during labs. Failure to participate fully and cooperatively will count against you; you could receive a one- or two-point deduction in your total course points. Since there are two pairings during the course, your total point reduction could be up to four points.

At the end of Assignment Three and at the end of the course you will be asked to individually submit a brief partner evaluation. This is required and due when indicated on the Course Schedule. As discussed in the Course Reference, you earn one point for a cursory evaluation, two points for an evaluation that is thorough or insightful, and no points if you turn an evaluation in late or not at all.

SEEK CLARIFICATION

Pair programming is shown to help, not hinder, your successful completion of the introductory course. It is important that you understand the processes and expectations up front so you can gain the most benefit. If you are unsure of any of the aspects of pair programming and how it is implemented in the course, see your instructor right away.

Pair Programming Guidelines — Course Staff Handout

[adapted from 26]

Many instructors assign duties for a course to teaching assistants, lab tutors, or both. This handout describes course staff responsibilities related to the pair programming aspects of a sample course. If you are viewing this document in pdf format, be aware that this file is also available as a modifiable Microsoft Word (.doc) at ***www.ncwit.org/pairprogramming.*** *Modify these instructions to suit your circumstances.*

SUPPORTING PAIR PROGRAMMING IN THE INTRODUCTORY COURSE

In CS1 students work with partners to complete programming work, using an approach called "Pair Programming." In pair programming, one student in the pair "drives," (types on the computer) while the other "navigates" (asks questions, makes suggestions, and points out errors). The partners switch roles every twenty minutes or so.

This guide describes the activities of course staff related to pair programming.

TEACHING ASSISTANT AND LAB TUTOR ROLES

If you are a teaching assistant you are responsible for forming and recording the pairing of students, ensuring pairs follow the pair programming model, and assisting pairs with their work. You will monitor how pairs perform and deal with problems that arise. Activities marked "TA" below apply particularly to teaching assistants. But, as teaching assistants also perform the duties of lab tutors, all other parts of the document apply to you as well.

If you are a lab tutor, your responsibilities are fewer. During lab sessions and while on duty outside of lab you answer partners' questions about their programming work and the hardware and software used in lab. All sections not marked "TA" describe your duties. But, in order to better understand other teaching roles in the course, read the sections specified for teaching assistants, too.

All course staff members are advised to read *Pair Programming in CS1—Sample Student Handout* in order to become familiar with pair programming and how it is implemented in this course. Also, they should view the video mentioned in the handout to ensure they understand the process and "do's and don'ts." The handout is a handy, common reference for you and the students in the course.

Address any questions to the instructor well ahead of the time you will first be in the lab.

(TA) Help students find partners, record partnerships. There may be several steps to grouping all students into pairs:

- Give students several days to choose partners. Some will quickly find a suitable partner. Others may take longer, "interviewing" one or more students before they settle on a partner.
- Help students who do not find a partner within several days. (Some students just find it hard to approach others, especially when they are new to computing courses or do not know anyone in the class.) One good approach is to gather together those students without partners and ask them to share their names, majors, year in school, roughly where they live (students living near each other can meet more easily), degree of previous programming experience, and any scheduling preferences. Encourage students to note which of their classmates seems like a likely partner. Once the introductions are complete, have students talk with others in the group whom they find potentially suitable. Several pairings should quickly result.
- If a student tries but cannot find a partner, you need to assign one. Try to pair the student with another student who has roughly the same skill level and programming experience and, if possible, lives reasonably close to the student.
- If there is one student remaining after pairs are formed, talk to colleagues or the instructor to see if students in other sections need a partner. If that fails, see if a pair in your section is willing to take on a third member. In a triad, two members navigate while one drives, and each takes turns as driver. Please make a team of three only if it cannot be avoided.
- If a student insists that she or he must work alone, tell the student to contact the instructor, whose permission is needed in order for a student to work singly. Pair programming is a course requirement so students cannot simply opt out.

» Record the student pairings, post a chart in the lab, and distribute the list to the instructor and other staff who may work with students.

Ensure students follow the pair programming model at all times. Pair programming means two students working as partners, with one person typing as the other watches and asks questions, makes comments, and helps to solve problems. They switch roles about every twenty minutes. While this way of working isn't familiar, it is important to student learning. Students may be reluctant to switch roles at first. For example, member A may type faster than member B, and so B will allow A to type—or A will insist on continuing to type, thinking the work will get done faster this way. (It probably won't, but the students do not have sufficient experience to know this.)

It's critical to get students into the pair programming habit from the very beginning, as it is hard to get them to follow the model once they've been "doing it their way" for several weeks. In particular, be sure they switch roles. Every twenty minutes or so, call out "It's time to switch," and then make sure partners do so.

Note that just "working together" in some informal way is not as beneficial as following the pair programming model with precision.

Monitor students to be sure they continue to do pair programming. Politely but firmly guide students who have strayed from the model practices back to consistent pair programming. It helps to smile while being firm.

(TA) If there are problems with a pair, deal with the problems as soon as possible. If problems persist after your intervention, seek advice from the instructor.

Remind students to prepare for lab. They can do this by reading and thinking about the assignment beforehand. They should be ready to work when they sit down at the computer. Remind students that being unprepared puts an unfair burden on their partner and delays the team's progress. Suggest pairs meet outside of lab to discuss and clarify the assignment before they begin to code it up.

(TA) Be available for private meetings with students. Tell students you are willing to meet with them individually to discuss any course-related matter. This signals to students that if they have a problem with their partner they can talk with you about it in confidence.

Be prepared to answer students' questions. You have to be "up" on lecture material and the assignments in order to help students with the material. It is a good idea to work the programming assignments yourself. This may not be much fun, but it's the only way you can get a feel for the difficulties and subtleties students will encounter.

Encourage students to ask questions—after the pair has tried to resolve them. Reassure students that difficulties are normal. Help foster a belief that they can complete the work and do well. Especially encourage students to ask their partner questions before coming to you; it often turns out that the partner will know the answer to a question, or the pair working together can quickly arrive at the answer. We want students to get used to relying first and foremost on each other.

Give students help, not answers. It is fine to tell people how to solve system-related problems, but you should lead them to their own solutions of problems related to the topics of the lab assignments. They should not become dependent on you; they should work with their partner, learn how to use manuals, read on-line documents and help files, and experiment. Ask them questions such as "What have you tried?", "What have you done so far?", "Where did things start going wrong?", "What are the possible issues?" If the problem is with understanding the assignment, perhaps say, "Let's read the assignment again and you tell me where it stops making sense." It will take you some time to learn these techniques, but it will be beneficial to you and the students you support in the future.

Be pleasant but ruthless about this. Students will beg you for a quick-fix answer, and it can seem easier for you to just give in than to point them towards finding the solution themselves. Nonetheless, resist the temptation, because students will come to expect you to fix their problems for them.

Try to help all who need assistance. Spread yourself around to everyone rather than concentrating on just a few pairs. If many people want your attention at once, you'll have to leave the current pair you are helping with a task to do independently while you make the rounds. Make sure each pair gets at least "first aid" help so they can get back to work; after that, feel free to spend more in-depth time with pairs that need it.

Appear available. You should circulate around and help students, not simply sit in a corner and wait for them to call on you. Ask every pair on occasion if all is going well; such querying gives less assertive students an opportunity to ask questions.

Address the problems of very experienced students. A pair of experienced students often thinks it can wait until the last minute to complete assignments; this may be true for the first assignment or two, but significant work takes substantial time, even for people who understand the concepts. More advanced students may not have experience following rigid specifications imposed by someone else (recreational computer users can just change the requirements), but our students have to solve hard problems rather than work around them. Remind students that paying attention to and following specifications is critically important, both in this class and in the profession.

Don't be too hard on yourself. Don't feel bad if you can't immediately identify a student's problem. While most of us can see problems quickly for the early, simpler assignments, this may lead students to expect (unrealistically) that we'll always be able to find and fix their problems at a glance. Of course, nobody can do that for complex assignments (such as the later ones in this class) and you need to explain this in lab and encourage students to think, experiment, use reasoning, and *use each other* to find and fix problems. It is okay to say you don't know, as long as you let students know that together you can find resolve difficulty by seeking advice of the instructor.

(TA) Score pair programming evaluations and evaluate students' performance as pair programming partners. Follow guidelines described in the course syllabus. If lab tutors are involved, encourage them to report to you about pairs they notice who are performing particularly well or poorly, giving a "heads up" about events you may not have noticed. Check out and confirm these observations for yourself before you include them in an evaluation of a student's or pair's performance. Do not rely on hearsay.

PAIR PROGRAMMING IS WORTH THE EFFORT

Pair programming benefits everyone. Hold yourself and your students to the pair programming model and you will see gains in students' learning and in their confidence as computer scientists.

NOTES:

Sample Pair Programming Questionnaire

[adapted from 27-29]

A short questionnaire provides feedback on a student's impressions of a student's pair programming partner, his or her own pair programming activity, and the pair programming process overall. If you are viewing this document in pdf format, be aware that this file is also available as a modifiable Microsoft Word (.doc) at ***www.ncwit.org/pairprogramming.*** *Modify the format and content of this to suit your conditions.*

DIRECTIONS

Answer each of the questions below and submit to your instructor or TA as advised. Be sure to give reasons for or elaborate on all of your answers; our goal is to understand the nature of your partner's participation in paired work, your own participation, and what you think about pair programming.

Do your best to be fair and objective. We will not share your scores and comments with your partner but you may do so if you choose. In fact, an open discussion may be very useful to you both.

Your Name: _____ **Your Partner's Name:** _____

1. **My partner's technical competency is (check one option to fill the blank):**

   Much better than mine ____

   Somewhat better than mine ____

   About the same as mine ____

   Somewhat less than mine ____

   Much less than mine ____

2. **My partner and I were... (check one option):**

   Very compatible ____

   Somewhat compatible ____

   Not at all compatible ____

For each of the following, respond about both yourself and your partner. Be sure to explain your answers.

3. **Arrived at scheduled meetings on time and ready to work. Notified partner about inability to attend a meeting or fulfill a responsibility.**

   You: Yes ____ No ____

   Your Partner: Yes ____ No ____

   Explain:

4. **Reviewed the assignment and preparatory materials before coming to the scheduled meetings, showing up either with specific questions or ready to contribute.**

   You: Yes ____ No ____

   Your Partner: Yes ____ No ____

   Explain:

5. **Cooperatively followed the pair programming model (rotating roles of driver and navigator, questioning and making observations as the navigator).**

   You: Yes ____ No ____

   Your Partner: Yes ____ No ____

   Explain:

6. **Contributed fully, fairly, and actively, to the best of ability, to the completion of the lab assignment.**

   You: Yes ____ No ____

   Your Partner: Yes ____ No ____

   Explain:

7. **Participation was professional and cooperative overall.**

   You: Yes ____ No ____

   Your Partner: Yes ____ No ____

   Explain:

Open-ended questions. Please supply thorough answers.

8. Describe the degree to which you have you found pair programming helpful in completing the programming work and learning about programming in general. Explain.

9. If you have previous experience programming alone, do you now prefer working in pairs? Why or why not?

10. Overall rating of yourself and your partner: put an X next to the rating that you most agree with.

You	Your Partner	Rating	Performance Category
		Excellent	Consistently performed A-level work, helped partner learn, and carried a fair share of the load
		Good	Consistently did what was expected, was well prepared, cooperative
		Satisfactory	Usually did what was expected, was somewhat prepared, generally cooperative
		Unsatisfactory	Seldom did what was expected, was unprepared, seldom cooperative
		No Show	Little to no participation

Other comments:

Sample Student Survey*

[adapted from 29]

Use this survey to gather data on students' pair programming experiences and opinions at the end of, and perhaps periodically during, the course. This instrument is suitable for performing data analysis, including along demographic dimensions. If you are viewing this document in pdf format, be aware that this file is also available as a modifiable Microsoft Word (.doc) at ***www.ncwit.org/pairprogramming.*** *Modify the format and content of this to suit your conditions.*

Circle the single word or phrase to indicate your response. You may skip questions that you do not wish to answer. Your participation is completely voluntary.

1.	How much experience with pair programming did you have before you began lab assignments for this class?	None	Only a little	A moderate amount	A great deal	
2.	When the class began, who knew more about programming?	My partner knew more	We had about equal knowledge	I knew more		
3.	How many pair programming partners did you work with during the course?	None	1	2	3 or more	
4.	How much help did your instructor or TA give you with finding a programming partner?	Not applicable, partners were assigned	None at all	Only a little	A moderate amount	Extensive
5.	How did having a partner affect the quality of your programs compared to what you would have written alone? The programs were...	Lower in quality	About the same quality	Higher in quality		

Please indicate your agreement or experience below:

6.	I wanted to change to a different pair programming partner.	Strongly Disagree	Disagree	Agree	Strongly Agree
7.	Having a partner made it easier to complete assignments.	Strongly Disagree	Disagree	Agree	Strongly Agree
8.	Having a partner made me feel more confident that the code was reliable.	Strongly Disagree	Disagree	Agree	Strongly Agree
9.	It was easy for me to find a pair programming partner.	Strongly Disagree	Disagree	Agree	Strongly Agree
10.	My pair programming partner helped me learn programming.	Strongly Disagree	Disagree	Agree	Strongly Agree

*© 2008 by Lecia Barker and Suzanne Schaefer. The authors give the user permission to modify the format and content of this to suit his/her course conditions.
To copy otherwise, republish, or redistribute requires permission of the authors.

11.	My partner contributed ideas towards writing a good program.	Never	Once or twice	Often	All the time
12.	It was helpful to discuss programming problems and solutions with a partner.	Strongly Disagree	Disagree	Agree	Strongly Agree
13.	My partner and I caught each other's mistakes.	Never	Once or twice	Often	All the time
14.	My partner and I were equally matched in terms of the pace at which we solved programming assignments.	Strongly Disagree	Disagree	Agree	Strongly Agree
15.	I found it easy to meet with my pair programming partner to work on lab assignments outside of our scheduled lab times.	Strongly Disagree	Disagree	Agree	Strongly Agree
16.	Having a partner is beneficial for learning to read another programmer's code.	Strongly Disagree	Disagree	Agree	Strongly Agree
17.	I spent about the same amount of time being the 'navigator' as I did being the 'driver.'	Strongly Disagree	Disagree	Agree	Strongly Agree
18.	I am comfortable being the 'driver.'	Strongly Disagree	Disagree	Agree	Strongly Agree
19.	I am comfortable being the 'navigator.'	Strongly Disagree	Disagree	Agree	Strongly Agree
20.	At this point in the class, my pair programming partner and I are evenly matched.	Strongly Disagree	Disagree	Agree	Strongly Agree
21.	I would recommend pair programming to other students.	Strongly Disagree	Disagree	Agree	Strongly Agree
22.	Pair programming leads to more success than individual programming.	Strongly Disagree	Disagree	Agree	Strongly Agree
23.	Pair programming should be part of every class that requires programming assignments.	Strongly Disagree	Disagree	Agree	Strongly Agree
24.	Pair programming helps me develop my teamwork skills.	Strongly Disagree	Disagree	Agree	Strongly Agree
25.	My TA is knowledgeable about pair programming.	Strongly Disagree	Disagree	Agree	Strongly Agree
26.	My TA likes pair programming.	Strongly Disagree	Disagree	Agree	Strongly Agree

*© 2008 by Lecia Barker and Suzanne Schaefer. The authors give the user permission to modify the format and content of this to suit his/her course conditions.
To copy otherwise, republish, or redistribute requires permission of the authors.

27. If there were one thing that I could change about my pair programming experiences, I would do the following: (Please be as specific as possible.)

To the Instructor: Modify these questions to reflect the kind of staff support your students receive during pair programming sessions.

28.	My lab tutors help me learn.	Strongly Disagree	Disagree	Agree	Strongly Agree
29.	My TAs help me learn.	Strongly Disagree	Disagree	Agree	Strongly Agree
30.	It was easy to get my questions answered during lab times.	Strongly Disagree	Disagree	Agree	Strongly Agree
31.	There were enough lab hours with the TA present.	Strongly Disagree	Disagree	Agree	Strongly Agree
32.	My TA knows the name of the students in my lab section.	Strongly Disagree	Disagree	Agree	Strongly Agree
33.	The person who answered most of my questions about the lab assignments was...	My pair programming partner	My TA	My lab tutor	Other
34.	It was easy for me to get my TA to answer my questions.	Strongly Disagree	Disagree	Agree	Strongly Agree
35.	It was easy for me to get my lab tutor to answer my questions.	Strongly Disagree	Disagree	Agree	Strongly Agree
36.	It was easy for me to get my pair programming partner to answer my questions.	Strongly Disagree	Disagree	Agree	Strongly Agree
37.	I am comfortable talking with my TA one-on-one.	Strongly Disagree	Disagree	Agree	Strongly Agree
38.	I am comfortable talking with my lab tutor one-on-one.	Strongly Disagree	Disagree	Agree	Strongly Agree
39.	I am comfortable talking with my professor one-on-one.	Strongly Disagree	Disagree	Agree	Strongly Agree
40.	I am comfortable talking with my pair programming partner one-on-one.	Strongly Disagree	Disagree	Agree	Strongly Agree

*© 2008 by Lecia Barker and Suzanne Schaefer. The authors give the user permission to modify the format and content of this to suit his/her course conditions.
To copy otherwise, republish, or redistribute requires permission of the authors.

41.	Overall, this course moves along at a good pace for me.	Strongly Disagree	Disagree		Agree	Strongly Agree
42.	During a given week, the average number of hours I spend on homework and lab for this class is...	Less than 1 hour	1–3 hours	4–6 hours	7–9 hours	10 or more hours
43.	I expected to spend fewer hours on homework and lab.	Strongly Disagree	Disagree		Agree	Strongly Agree
44.	The assignments are better suited for some of the more experienced students than they are for me.	Strongly Disagree	Disagree		Agree	Strongly Agree
45.	How confident are you that you will complete your chosen major?	Not at all confident	Slightly confident		Moderately confident	Very confident

46. The aspect of this class that has been the *least* challenging for me: (please be as specific as possible)

47. The aspect of this class that has been the *most* challenging for me: (please be as specific as possible)

48. If I could change any one aspect of this class, this is what I would do: (please be as specific as possible)

49. What year were you born? 19_____

50. Are you: Male Female

51. What is your academic year?

a. First year b. Second year c. Third year d. Fourth year e. Fifth year senior

*© 2008 by Lecia Barker and Suzanne Schaefer. The authors give the user permission to modify the format and content of this to suit his/her course conditions.
To copy otherwise, republish, or redistribute requires permission of the authors.

52. What is your major?

53. The following describes the likelihood I will complete my major:	Not at all likely	Somewhat likely	Very likely	Other (please specify):

54. Which category or categories best describe you? (Circle all that apply.)

Black/African-American	Native American
Southeast Asian	Pacific Islander
Chicano / Latino / Hispanic	White/Caucasian
Indian Subcontinent	Other: _____
Middle Eastern	

Thank you for taking the time to complete this questionnaire. Your responses will help us to improve our students' experience in the program. Please use the space below to tell us about any issues/questions we may have overlooked or ways we can improve this survey.

*© 2008 by Lecia Barker and Suzanne Schaefer. The authors give the user permission to modify the format and content of this to suit his/her course conditions.
To copy otherwise, republish, or redistribute requires permission of the authors.

Resources and Acknowledgments

Selected materials include research papers, course materials, videos, and web resources grouped by topic.

INDEX

A. Pair programming overview

B. Pair programming in introductory computing courses

C. Pair programming in the classroom

D. Compatible pairs

E. Lab exams

F. Preparing course staff

G. Evaluating programming pairs

H. Resource lists on the Web

I. Pair programming materials for students

A. PAIR PROGRAMMING OVERVIEW

Pair programming is done in programming courses, but it is also used in other contexts, particularly in the software development industry. These sources discuss pair programming generally:

[1] Pair Programming. (2008). In *Wikipedia, the free encyclopedia,* http://en.wikipedia.org/wiki/Pair_programming.

[2] Lui, K.M. & Chan, K. C. (2006). Pair programming productivity: Novice-novice vs. expert-expert. *International Journal of Human-Computer Studies,* 64(9), 915-925.

[3] Williams, L. & Kessler, R. (2003). *Pair Programming Illuminated.* Reading, MA: Addison-Wesley Professional.

[4] Cockburn, A. & Williams, L. (2001). The costs and benefits of pair programming. In G. Succi & M. Marchesi (Eds.), *Extreme Programming Examined.* (pp. 223-243). Boston, MA: Addison-Wesley Longman.

B. PAIR PROGRAMMING IN INTRODUCTORY COMPUTING COURSES

Pair programming is quite effective in first programming courses. These sources discuss what's been learned from the study of various implementations of pair programming in introductory computing courses, particularly CS1:

[5] Jacobson, N.M. & Schaefer, S.K. (2008). Pair programming in CS1: Overcoming objections to its adoption. *ACM SIGCSE Bulletin, 40*(2), 93-96.

[6] Braught, G., Eby, L.M., & Wahls, T. (2008). The effects of pair-programming on individual programming skill. *ACM SIGCSE Bulletin, 40*(1), 200-204.

[7] Simon, B. & Hanks, B. (2008). First-year students' impressions of pair programming in CS1. *Journal on Educational Resources in Computing (JERIC), 7*(4), 1-28.

[8] Howard, E. (2006-2007). Attitudes on using pair-programming. *Journal of Educational Technology Systems, 35*(1), 89-103.

[9] Hanks, B. (2005). Student performance in CS1 with distributed pair programming. *ACM SIGCSE Bulletin, 37*(3), 316-320.

[10] Williams, L., Yang, K., Wiebe, E., Ferzli, M., Miller, C. (2002, November). *Pair programming in an introductory computer science course: Initial results and recommendations.* Paper presented at the OOPSLA Educator's Symposium, Seattle, WA.

[11] Williams, L., Wiebe, E., Yang,K., Ferzli, M., Miller, C. (2002). In support of pair programming in the introductory computer science course. *Computer Science Education, 12*(3), 197-212.

[12] McDowell, C., Werner, L., Bullock, H. & Fernald, J. (2002). The effects of pair-programming on performance in an introductory programming course. *Proceedings of the 33rd SIGCSE Technical Symposium on Computer Science Education,* 38-52.

[13] Bevan, J., Werner, L., McDowell, C. (2002). Guidelines for the use of pair programming in a freshman programming class. *Proceedings of the 15th Conference on Software Engineering Education and Training.*

C. PAIR PROGRAMMING IN THE CLASSROOM

These sources examine and discuss pair programming as used in programming courses other than "the first course", or across several courses. Number 14 is a video that explains pair programming to students.

[14] Williams, L. *Guidelines for Implementing Pair Programming in the Classroom,* a video to be released May, 2009. Search http://agile.csc.ncsu.edu/pairlearning/.

[15] Williams, L. *Common Perceptions and Myths about Pair Programming,* a video to be released May, 2009. Search http://agile.csc.ncsu.edu/pairlearning/.

[16] Williams, L. *An Introduction to Pair Programming for Students,* a video to be released May, 2009. Search http://agile.csc.ncsu.edu/pairlearning/.

[17] Williams, L. (2007). Lessons learned from seven years of pair programming at North Carolina State University. *ACM SIGCSE Bulletin, 39*(4), 79-83.

[18] McDowell, C., Werner, L., Bullock, H. E., & Fernald, J. (2006). Pair programming improves student retention, confidence, and program quality. *Communications of the ACM, 49*(8), 90-95.

[19] Hanks, B. (2006). Student attitudes toward pair programming. *ACM SIGCSE Bulletin 38*(3), 113-117.

[20] Hanks, B., McDowell, C., Draper, D., & Krnjajic, M. (2004). Program quality with pair programming in CS1. *Inroads, 36*(3),176-180.

[21] Werner, L. L., Hanks, B., & McDowell, C. (2004). Pair-programming helps female computer science students. *Journal on Educational Resources in Computing (JERIC), 4*(1).

[22] Williams, L. & Upchurch, R. L. (2001). In support of student pair-programming. *ACM SIGCSE Bulletin (33)*1, 327-331.

D. COMPATIBLE PAIRS

Compatibility is essential to an effective and enjoyable pair programming experience; this paper focuses on factors that affect compatibility:

[23] Williams, L., Layman, L., Osborne, J., & Katira, N. (2006). Examining the compatibility of student pair programmers. *Proceedings of the conference on AGILE 2006,* 411-420.

E. LAB EXAMS

Lab exams for individual students are an excellent alternative to assignments in which a pair's efforts are graded. They remove the concern that grades of pair work do not accurately reflect students' individual competencies. These publications discuss various methods of lab examination:

[24] Barros, J. P., Estevens, L., Dias, R., Pais, R., Soeiro, E. (2003). Using lab exams to ensure programming practice in an introductory programming course. *ACM SIGCSE Bulletin 35*(3),16-20.

[25] Jacobson, N. (2000). Using on-computer exams to ensure beginning students' programming competency. *ACM SIGSCE Bulletin* (*32*)4, 53-56.

F. PREPARING COURSE STAFF

The sample teaching assistant and lab tutor handouts are based on this document:

[26] Jacobson, N. (2008, Winter). *Duties and Responsibilities of ICS 21 Course Staff,* Unpublished, available from the author.

G. EVALUATING PROGRAMMING PAIRS

These pair programming student questionnaires and survey served as the basis of the sample questionnaire and survey:

[27] Williams, L. *PairEval Team Member Evaluation,* http://agile.csc.ncsu.edu/pairlearning/paireval/peereval1.jpg

[28] Jacobson, N. (2008). *Pair Programming Partner Evaluation and Choosing Your Next Partner,* www.ics.uci.edu/~jacobson/ics21/LabManual/08A-PartnerSwapAndEval.html

[29] Barker, L. & Schaefer, S. (2008). *Bren School of Information and Computer Sciences Undergraduate Student Survey.* Unpublished.

H. RESOURCE LISTS ON THE WEB

These pair programming researchers have prepared Web pages that list additional sources of information about pair programming:

[30] Laurie Williams, Computer Science Department, North Carolina State University. http://collaboration.csc.ncsu.edu/laurie/pair.html

[31] Charlie McDowell and Linda Werner, Computer Science Department, UC Santa Cruz. www.soe.ucsc.edu/~charlie/projects/pairprogramming

[32] Brian Hanks, Computer Science Information Systems, Fort Lewis College, Colorado. http://faculty.fortlewis.edu/hanks_b/

I. PAIR PROGRAMMING MATERIALS FOR STUDENTS

Videos:

[33] Williams, L. *An Introduction to Pair Programming for Students,* a video to be released May, 2009. Search http://agile.csc.ncsu.edu/pairlearning/.

[34] Werner, L. and Denner, J. *A video on the right way and the wrong way to do pair programming.* www.cse.ucsc.edu/~charlie/projects/pairprogramming/video

Reading:

[35] Williams, L. A. & Kessler, R. R. (2000). All I really need to know about pair programming I learned in kindergarten. *Communications of the ACM, 43*(5), 108-114.

Handouts from actual courses, describing pair programming:

[36] Arayametee, S. (2008). *CS302 Pair Programming Guidelines.* http://pages.cs.wisc.edu/~cs302-1. Click on Programming Assignments, scroll to Summary of Pair Programming.

[37] Jacobson, N. (2008). *Pair Programming Guidelines.* http://www.ics.uci.edu/~jacobson/ics21/LabManual/02A-PairProgramming.html

[38] Kay, D. G. (2007). Pair Programming Guidelines. http://www.ics.uci.edu/~kay/courses/i41/hw/pair.html

[39] McDowell, C. *Pair Programming Guidelines/Rules.* http://www.soe.ucsc.edu/~charlie/projects/pairprogramming/PairProgrammingGuidelines.html

[40] McKinley, K. S. (2005). Pair Programming in *CMPSCI 305j Introduction to Computing Syllabus.* www.cs.utexas.edu/users/mckinley/305j/syllabus.html

[41] Parrish, E. (2002). *Pair Programming for Homework Assignments.* www.edparrish.com/cis12/f02/pairprog.asp

Student comments about their pair programming experiences, with instructor commentary:

[42] Kay, D.G. *Pair Programming Evaluations.* www.ics.uci.edu/~kay/courses/i41/hw/evalcomments.html

ACKNOWLEDGMENTS

Pair Programming-in-a-Box was developed by NCWIT in collaboration with the Ada Byron Research Center, Donald Bren School of Information and Computer Sciences, University of California–Irvine.

Pair Programming-in-a-Box was created by Norman Jacobson, University of California, Irvine and edited by Jane Krauss, NCWIT. Thanks to the following people for their significant contributions: David Kay, University of California, Irvine; Laurie Williams, North Carolina State University; Vicki Almstrum, University of Texas at Austin; Suzanne Schaefer, University of California, Irvine; Charlie McDowell, University of California, Santa Cruz; Lecia Barker, University of Texas at Austin; and Joanne Cohoon, University of Virginia.

Evaluating *NCWIT Pair Programming-in-a-Box: The Power of Collaborative Learning*

In order to continually improve its programs, NCWIT relies on those who use them to give critical feedback. After reviewing this Program-in-a-Box, please take a moment to respond to a short initial impressions survey about the contents and usefulness of *Pair Programming-in-a-Box*.

To access the survey, visit **www.ncwit.org/pairprogramming,** and click on the button to the right of your screen.