Student Learning Assurance Report

Department of Computer Science
College of Arts and Sciences
University of San Francisco

November 10, 2012

1 Introduction

In the spring of 2012, the undergraduate program in computer science assessed the following learning outcomes:

1. Use programming constructs (iteration, functional decomposition, recursion) in at least two different programming languages.

4. Implement standard data structures and algorithms (binary search trees, sorting, etc.) in at least two different programming languages.

The following faculty were involved in the evaluation of these outcomes: Sophie Engle, Patricia Francis-Lyon, Terence Parr, and Sami Rollins.

2 Outcome 1

Among the courses we offered last spring, the following courses had either moderate or comprehensive coverage of outcome 1:

- CS 110, Introduction to computer science I, Profs. Engle and Francis-Lyon.
- CS 112, Introduction to computer science II, Prof. Rollins.
- CS 212, Software development. Prof. Engle.
- CS 345, Programming language paradigms. Prof. Parr
- CS 490, Senior team project. Prof. Francis-Lyon.

The rubrics for evaluating outcome 1 are

- Unacceptable: unable to solve basic problems (e.g., searching/sorting).
- Acceptable: can solve basic problems (e.g. sorting/searching) in two languages.

- Exemplary: can solve more complex problems in multiple ways (e.g., recursion and iteration).

2.1 CS 110-01, Prof. Engle

Prof Engle chose the following questions from her final exam

5a. What is the output of the following code? Create the same list using a nested for loop.

```
words = [ i + j for i in ("b", "h", "m") for j in ["eh", "oo"] ]
print words
```

6b. Write a Python function that uses recursion to calculate and return the factorial of a number. The factorial function is defined as follows:

$$\begin{aligned} 0! &= 1 \\ n! &= n(n-1)! \end{aligned}$$

10. Read the file `test.py` one line at a time. For each line read, write the line number and that line to the file output.txt. For example, suppose `test.py` contains the following lines:

```
name = raw_input("Enter your name: ")
print "Hello " + name + "!"
```

The text in output.txt should be:

```
1 name = raw_input("Enter your name: ")
2 print "Hello " + name + "!"
```

Some starter code has been provided for you below.

```
# current line number
num = 0
# input and output files
infile = open("test.py", "r")
outfile = open("output.txt", "w")
# read one line at a time
# write current line number and current line to output file
. . .
# close files
infile.close()
outfile.close()
```

For each student, she totalled his or her scores on the three questions. Out of a total of 18 possible points, she determined that a score of 18 was exemplary, a score in the range 7–17 was acceptable, and a score in the range 0–6 was unacceptable. In a class of 23 students

- 5 did unacceptable work,
- 15 did acceptable work, and
- 3 did exemplary work.

2.2 CS 110-02, Prof. Francis-Lyon

Prof. Francis-Lyon evaluated the students' performance on the following questions:

2. Assume `lst` is a non-empty list of integers. Write a Python function that returns the index of the minimum element of list `lst`
4. Write function `removeLowest(lst)` so that it doesn't do any calculating itself, but merely calls the functions that you wrote for problems 2 and 3. You will get full credit for 4 regardless of whether 2 and 3 are correct as long as they are used and called correctly. Assume `lst` is a non-empty list of integers.
9. Write a recursive function that will return the sum of the squares of the elements of a list of numbers. You must solve this with recursion, you may not use a loop.

For each question, she judged the student's work as follows.

- Unacceptable: skipped question or major errors with hand-written algorithm.
- Acceptable: minor errors with hand-written algorithm that would have been debugged quickly if typed in and run in Python.
- Exemplary: hand-written algorithm is completely correct and free of error.

The counts are as follows:

- Question 2: 0 unacceptable, 5 acceptable, 11 exemplary
- Question 4: 1 unacceptable, 4 acceptable, 11 exemplary
- Question 9: 4 unacceptable, 8 acceptable, 4 exemplary

In a class of 16 students.

- 3 did unacceptable work,
- 8 did acceptable work, and
- 5 did exemplary work.

2.3 CS 112-01, Prof. Rollins

Prof. Rollins evaluated the students' performance on the basis of the following question from the final exam:

- Implement a LinkedList method `count`. The method takes as input a `String` representing list data and will return the number of times the `String` object appears in the list. for full credit implement this method using recursion. For a maximum of 8/10 points, implement the method using iteration. You may use a helper method. The header of the method is as follows:

```
public int count(String);
```

In a class of 11 students

- 3 did unacceptable work,
- 4 did acceptable work, and
- 4 did exemplary work.

2.4 CS 212-01, Prof. Engle

Prof. Engle evaluated the students' performance on each of the following projects:

- Project 2: Build an inverted index with partial search capability.
- Project 3: Build a thread-safe inverted index with multithreaded partial search capability.

Each student's work was scored on each project in the range 0–1 (fractional scores were possible). After an individual's scores were added, his or her work was judged unacceptable if the total was less than 1, acceptable if the total was greater than or equal to 1 but less than 2, and exemplary if the total was 2. In a class of 18 students

- 7 did unacceptable work,
- 3 did acceptable work, and
- 8 did exemplary work.

2.5 CS 345-01, Prof. Parr

Prof Parr evaluated the students' work on the basis of three programming projects:

1. Build a read-evaluate-print-loop interface for Java code.

2. Build a sampling profiler and a tracing profiler in Java.

3. Build a mark-and-compact garbage collector in C.

Each student's work was judged inadequate, adequate, or exemplary on the basis of the student's score. The 20 student's scored as follows.

- Project 1: 2 unacceptable, 4 acceptable, 14 exemplary
- Project 2: 3 unacceptable, 0 acceptable, 17 exemplary
- Project 3: 7 unacceptable, 4 acceptable, 9 exemplary

In a class of 20 students

- 7 did unacceptable work,
- 4 did acceptable work, and
- 9 did exemplary work,

3 Outcome 4

Among the courses we offered last spring, the following courses had either moderate or comprehensive coverage of outcome 4:

- CS 112, Introduction to computer science II, Prof. Rollins.
- CS 212, Software development. Prof. Engle.
- CS 345, Programming language paradigms. Prof. Parr

The rubrics for evaluating outcome 4 are

- Unacceptable: unable to implement moderately complex data structures (e.g., binary search trees and complex linked lists) in two programming languages.
- Acceptable: able to implement moderately complex data structures (e.g., binary search trees and complex linked lists) in two programming languages.
- Exemplary: able to implement complex data structures and algorithms (e.g., B+ trees, complex graphs) in two programming languages.

3.1 CS 112-01, Prof. Rollins

Prof. Rollins evaluated the students' performance on the basis of their answers following question from the final exam.

- Implement a LinkedList method `sublist`. The method takes as input an `int` representing an index and will create a sublist beginning at the given index by removing index Nodes fromt the beginning of the existing list. Given the list `head->A->B->C->D->E` `sublist(0)` would result in the same list. `sublist(2)` would result in the `head` now referring to the Node containing C. sublist(4) would result in the `head` now referring to the Node containing E. `sublist(5)` would result in an `IndexOutOfBoundsException`. You may *not* assume that a `remove` method exists. The header of the method is as follows:

```
public void sublist(int index);
```

In a class of 11 students

- 0 did unacceptable work,
- 6 did acceptable work, and
- 5 did exemplary work.

3.2 CS 212-01, Prof. Engle

Prof. Engle evaluated the students' performance on each of the following projects:

- Project 2: Build an inverted index with partial search capability.
- Project 3: Build a thread-safe inverted index with multithreaded partial search capability.

Each student's work was scored on each project in the range 0–1 (fractional scores were possible). After an individual's scores were added, his or her work was judged unacceptable if the total was less than 1, acceptable if the total was greater than or equal to 1 but less than 2, and exemplary if the total was 2. In a class of 18 students

- 7 did unacceptable work,
- 3 did acceptable work, and
- 8 did exemplary work.

3.3 CS 345-01, Prof. Parr

Prof Parr evaluated the students' work on the basis of three programming projects:

1. Build a read-evaluate-print-loop interface for Java code.
2. Build a sampling profiler and a tracing profiler in Java.
3. Build a mark-and-compact garbage collector in C.

Each student's work was judged inadequate, adequate, or exemplary on the basis of the student's score. The 20 student's scored as follows.

- Project 1: 2 unacceptable, 4 acceptable, 14 exemplary
- Project 2: 3 unacceptable, 0 acceptable, 17 exemplary
- Project 3: 7 unacceptable, 4 acceptable, 9 exemplary

In a class of 20 students

- 7 did unacceptable work,
- 4 did acceptable work, and
- 9 did exemplary work,

4 Conclusions

At this time we are still studying the results outlined in this report. Hence it is too early to make any decisions about how we will modify our curricula to improve student learning outcomes.