Summative Assessment of Baccalaureate Computer Science: The Graduating Senior Test - Version 2

The Seidenberg Faculty
About this issue:

This report archives the summative assessment test that superseded the one appearing in the previous issue. Instructions for scoring are at the end.

The test in the previous issue was denounced by the ABET accreditation examiners who visited us in Fall 2006. Its inadequacy lay in the fact that it was not aligned to the eleven learning outcomes claimed, at that time, by the BS in CS program. To document the effectiveness of the program and to identify areas for strengthening ("continuous improvement" is part and parcel to assessment), the graduating senior test needed to produce a score specific to each and every outcome. This included the technical outcomes, such as "an ability to analyze a problem and to identify and define the computing requirements appropriate to the solution" and "an ability to apply mathematical foundations, algorithmic principles, and computer science theory in the modeling and design of computer-based systems in a way that demonstrates comprehension of the tradeoffs involved in design choices." It also included soft outcomes, such as "an ability to function effectively on teams to accomplish a common goal," "an understanding of professional, ethical, and social responsibilities," and "an ability to communicate effectively with a range of audiences."

The faculty was stymied by the edumatic assignment from ABET. Many of the outcomes were of such an amorphous, high-level nature that it was difficult to imagine "hit the nail on the head" test items. In fact, their assessment seemed at odds with admonitions of proper practice: to focus on clear, concrete abilities that can be demonstrated (e.g. problems one can solve, processes one can diagram, algorithms one can recall and explain, etc.).

The faculty did what it had to do, and the test herein was administered to the CS majors in the class of 2007 and then to the classes of 2008 and 2009. There is one section of the test for each program outcome; hence eleven sections. The outcome to which each section applies is stated in the test.

If you, the reader, have doubts about the validity of various sections, let me simply say welcome to the club. Everyone thought it farcical, for instance, to suppose that three common-sense questions could meaningfully indicate how well students were communicating on technical concepts and issues, orally and in writing, to both peers and lay people. Clearly, the present instrument is not the end of the summative assessment story.

Despite its imperfections, the present instrument was an accomplishment. Thanks are due to Mary Courtney who, as Chairperson of the Computer Science Curriculum Committee, spearheaded the effort and mobilized the faculty. Also, huge amounts of help came from Andrea Cotoranu, the Seidenberg School's Associate Director of Assessment (informally, our ever-accessible, always encouraging, nothing-is-too-much-to-ask assessment guru).

Allen Stix, Editor
Summative Assessment for the Computer Science Major

The purpose of this test is not to evaluate you, but to assess the effectiveness of our program. The results will help us to see where coverage needs strengthening.

Please try your best because these test results are important. The entire CS faculty and staff thank you!

Section A: applying a knowledge of computing and mathematics appropriate to the discipline

1. **false**
   - For small data sets, an "n-squared" sorting algorithm (e.g. the bubble sort) may execute faster than an alternate n-lg-n algorithm (e.g. the quicksort).

2. **false**
   - In Java 5.0 and later, the execution time of an algorithm can be measured in greater precision than **thousandths of a second**.

3. **false**
   - For an exact solution to the traveling sales person problem or a problem of similar time complexity, scaling-up the size by a factor of 10 (e.g. from 5 cities to 50) increases the processing effort by a factor of roughly 1000 (i.e. on the order of n', where n is the size of the scale-up).
4. What is the time complexity of the following segment of code?

```java
for (int i = 0; i < n; i++)
{
    for (int j = i; i < n; j++)
    {
        //processing without iteration or selection
    }
}
```

a) $O(c)$
b) $O(N)$
c) $O(N \lg N)$
d) $O(N^2)$
e) none of the above
f) impossible to determine based on what is shown
5. Each of five cities is connected to each of the others with a direct highway. How many highways are there?

The numerical answer as well as the general formula can be derived by inspection from the adjacency matrix.

Sketch the matrix, circle "the location of the answer," and give the general formula.

<table>
<thead>
<tr>
<th></th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
<th>E</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>B</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>C</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>D</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>E</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Answer:

Name the cities A, B, C, D, and E.

The highways are undirected edges, which means that the same highway that connects city i to city j connects city j to city i.

The result is that each highway appears twice in the adjacency matrix, above the main diagonal and in the mirror-image position below.

No entries are on the main diagonal because no highway connects a city to itself.

<table>
<thead>
<tr>
<th></th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
<th>E</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>B</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>C</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>D</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>E</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

These 10 entries represent each highway once and only once.

The adjacency matrix has $5 \times 5 = 25$ elements. Subtracting the main diagonal, which consists of one element on each row, leaves $25 - 5 = 20$ elements. Because each highway is represented twice, there is space for $20/2 = 10$ highways. The isomorphism between the adjacency matrix and physical geography indicates that 10 highways are needed to connect each of the five cities directly with each of the others. In general:

$$\text{numOfHighways} = (\text{numOfCities}^2 - \text{numOfCities}) / 2$$
Section B: analyzing a problem and identifying and defining the computing requirements appropriate to its solution

1. Which is true of CRC cards?
   a) CRC cards are an outgrowth from the IBM punch card used in early electronic data processing systems.
   b) CRC cards are used in conjunction with role playing to identify the classes and API each class will need to offer as "servers" in a software system.
   c) CRC cards define an effective method for requirements capture but this methodology requires a solid understanding of class derivation hierarchies.
   d) Techniques based on CRC cards harness the knowledge of domain experts and are equally effective for procedural and object-oriented analysis and design.

2. What is the UML?
   a) A programming language for developing prototypes of object-oriented software.
   b) A variation of the same approach to software construction as extreme programming and agile development.
   c) An updated form of flow charting chiefly used for diagramming sequence.
   d) A set of graphical notations that convey information pertinent to the analysis and design of object-oriented software systems.

3. True false When declaring exceptions, use checked exceptions for abnormal occurrences from which recovery is conceivable. Use unchecked exceptions for problems of such severity that proceeding is impossible.
Section C: designing, implementing, and evaluating a computer-based system, process, component, or program to meet desired needs

1. **30** What is the value of the postfix expression below?

\[ 2 \ 3 \ 4 \ + \ 6 \ 2 \ - \ * \ + \]

Questions 2-5 pertain to implementing a stack. Mark an item false if any detail is untrue or inaccurate.

The API for the `java.util.Stack` class includes the following public instance methods:

```java
public boolean empty();
public Object pop();
public Object push(Object obj);
```

2. **true** false Suppose you are building a program to evaluate postfix expressions such as the one above. Java's `Stack` class pushes and pops `Objects`, but your program needs to work with numeric primitives. Despite this, a `java.util.Stack` object may be harnessed to serve as the LIFO storage structure for operands.

3. **true** false There is no logical limit to the number of `Objects` that can be pushed onto a `java.util.Stack` object.

4. **true** false An object of type `java.util.Stack` may be used whenever an application program requires a stack.

5. true **false** While more than one stack at a time within a single program is rarely needed, a limitation on the utility of `java.util.Stack` is that only one instance can be active in the name space.
Section D: functioning effectively on teams to accomplish a common goal

1. **true** false  Teams work more effectively when one member has decision-making power over the group's activities and takes responsibility for communicating, coordinating, monitoring, working out compromises, and supervising.

2. **true** false  In a small, informal, task-oriented group; it is important that the individual who assumes the lead is the one who is best liked.

3. **true** false  Little disagreement over new ideas among team members is an indicator that the team is working effectively toward its objective and that everyone feels comfortable interpersonally.
Section E: understanding professional, ethical and social responsibilities

1. true false As software engineering standards, product liability, standards for protecting data, and the like are increasingly prescribed by law; professional ethics (the practitioner's sense of right and wrong) is of diminishing importance.

2. true false Two strongly ethical individuals will ultimately agree on the which of two different courses of conduct constitutes "the high road."

3. What is deontology?

   a) an approach to ethical reasoning founded on a sense of duty to virtues that are accepted as good in and of themselves such as honesty, fair play, respect for privacy, freedom, truth, proper compensation, and beneficence (the obligation to help other people improve their lives)

   b) an approach to ethical reasoning founded on choosing action the will result in the greatest good and the least harm relative to all affected

   c) an approach to ethical reasoning founded on choosing action that will result in the greatest benefit to oneself (the person or organization taking the action)

   d) an approach to ethical reasoning founded on ferreting out the facts to understand the situation
Section F: communicating effectively with a range of audiences

1. **true** false  In preparing an informative oral presentation, it is important to consider the background of the audience (what they know) and what they wish to learn from you.

2. **true** false  In delivering an informative oral presentation, it is important to tell jokes to retain audience attentiveness.

3. **true** false  Even with spelling and grammar tools, it remains essential to re-read what you have written to be sure that nothing can be misunderstood, that your information is correct and complete, and that the tone will elicit a favorable response.

"Good writing is bad writing that was rewritten."
(Marc H. Raibert)
Section G: analyzing the impact of computing on individuals, organizations and society, including ethical, legal, security and global issues.

1. In what sense is an American corporation's decision to develop software offshore ethically justifiable even though it deprives work to American programmers?
   a) Offshore outsourcing is ethically reprehensible no matter how one looks at it.
   b) The quality of work by foreign software engineers is known to be better.
   c) The corporation is responsible to its share-holders to operate as profitably as possible in accordance with all legal means.
   d) Offshore outsourcing is not an issue amenable to ethical analysis.

2. (true) false Prognostications on the social impacts of technology are as likely to be wrong as they are to be right. One example is the prediction that the Internet would lead to more reasoned political opinions as individuals would seek information across a spectrum of sites. What actually happens is that individuals do not seek unbiased information. They go to sites that reinforce positions they already hold.

3. A survey at Pace finds that owning a laptop correlates with a higher grade point average. The link could be that having a laptop "on one's person" enables the more productive use of small blocks of time. However, the responsible mechanism may just as well be socioeconomic status (SES): laptop ownership might be a concomitant of the same economic well-being known to confer a scholastic advantage.

Assuming a stratified random sample consisting of 100 higher SES students and 100 lower SES students, which data would tend to rule out socioeconomic status as the explanatory mechanism? The data, four tables, are on the following page.
a) Higher Socioeconomic Status  
<table>
<thead>
<tr>
<th></th>
<th>QPA</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Hi</td>
<td>Lo</td>
</tr>
<tr>
<td>yes</td>
<td>45</td>
<td>15</td>
</tr>
<tr>
<td>no</td>
<td>25</td>
<td>15</td>
</tr>
<tr>
<td></td>
<td>70</td>
<td>30</td>
</tr>
</tbody>
</table>

b) Higher Socioeconomic Status  
<table>
<thead>
<tr>
<th></th>
<th>QPA</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Hi</td>
<td>Lo</td>
</tr>
<tr>
<td>yes</td>
<td>70</td>
<td>20</td>
</tr>
<tr>
<td>no</td>
<td>8</td>
<td>2</td>
</tr>
<tr>
<td></td>
<td>78</td>
<td>22</td>
</tr>
</tbody>
</table>

c) SES  
<table>
<thead>
<tr>
<th></th>
<th>Laptop</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>yes</td>
<td></td>
<td></td>
</tr>
<tr>
<td>higher</td>
<td>90</td>
<td>10</td>
</tr>
<tr>
<td>lower</td>
<td>10</td>
<td>90</td>
</tr>
<tr>
<td></td>
<td>78</td>
<td>22</td>
</tr>
</tbody>
</table>

d) QPA  
<table>
<thead>
<tr>
<th></th>
<th>Laptop</th>
</tr>
</thead>
<tbody>
<tr>
<td>yes</td>
<td></td>
</tr>
<tr>
<td>Hi</td>
<td>75</td>
</tr>
<tr>
<td>Lo</td>
<td>25</td>
</tr>
<tr>
<td></td>
<td>100</td>
</tr>
<tr>
<td>no</td>
<td></td>
</tr>
<tr>
<td></td>
<td>53</td>
</tr>
<tr>
<td></td>
<td>47</td>
</tr>
<tr>
<td></td>
<td>100</td>
</tr>
<tr>
<td></td>
<td>128</td>
</tr>
<tr>
<td></td>
<td>72</td>
</tr>
</tbody>
</table>
Section H: recognizing the need for, and an ability to, engage in continuing professional development

1. The computer science curriculum of twenty years ago most likely did not include:
   a) Java or C++
   b) browsers, HTML, or the HTTP protocol
   c) frameworks for graphical user interfaces
   d) extreme programming or agile techniques for software development
   e) all of the above

2. true  false  Computer science and software development are different from information technology as applied by organizations. Life-long learning is much less an issue for individuals not working in applied IT.

3. true  false  The fundamental principles underlying computing do not change rapidly. These are the focus of a baccalaureate that supplies a durable knowledge structure. In contrast, professional work entails learning about a never-ending succession of new tools, techniques, products, and services.
1. It has always been possible for "new code to run old code." This is exemplified by methods such as `Math.sqrt()`.

Polymorphism allows "old code to run new code."

Given the abstract `Shape` class and the concrete `Circle` class that extends it directly below:

```java
public abstract class Shape
{
    public abstract double getArea();
    public abstract double getPerimeter();
}

public class Circle extends Shape
{
    private double radius;

    public Circle(double radius) //constructor
    {
        this.radius = radius;
    }

    public double getArea()
    {
        return Math.PI * (radius*radius);
    }

    public double getPerimeter()
    {
        return Math.PI * (2*radius);
    }
}
```
class Main
{
    public static void main(String[] args)
    {
        Shape shape = new Circle(1.0);
        // "old code" that should work with "new" objects
        double area = shape.getArea();
        double perimeter = shape.getPerimeter();
        double ratio = area/perimeter;
        System.out.println("ratio = " + ratio);
    }
}

1.a. Suppose that the Circle class is "old" but that the Square class is a brand new descendent of Shape.

i) Change the statement on line 5 so that a Square object is instantiated with a side length of two:

    Shape shape = new Square(2.0);

ii) On line 8 and line 9 make whatever change(s) are needed to get object's area and perimeter. Show these lines here:

    no change needed

    no change needed
1.b. Write the code for this **Square** class:

**Answer:**

```java
public class Square extends Shape {
    private double sideLength;

    public Square(double sideLength)    //constructor
    {
        this.sideLength = sideLength;     //argument must match 1.a.i.
    }

    public double getArea()              //what's important is method's
    {
        return sideLength * sideLength;  //presence and signature
    }

    public double getPerimeter()         //what's important is method's
    {                                     //presence and signature
        return 4 * sideLength;
    }
}
```
2. Arrays of objects may be sorted using a `java.util.Comparator` object and the version of `java.util.Arrays.sort()` that accepts the array as its first argument and the `Comparator` object as its second.

An object of type `java.util.Comparator` is an instance of a class that implements the `java.util.Comparator` interface, shown below:

```java
interface java.util.Comparator
{
    int compare(Object objA, Object objB);
}
```

```
compare(a, b) returns:
{
-1 when a < b
0  when a == b
1  when a > b
}
```

Given the `Rectangle` class directly below. Which implementation of the `CompareByArea` class (on the next page) will work best to enable arrays of `Rectangles` to be sorted by their areas?

```java
class Rectangle
{
    private int length, height;

    public Rectangle(int length, int height)
    {
        this.length = length;
        this.height = height;
    }

    public int getLength()
    {
        return length;
    }

    public int getArea()
    {
        return height * length;
    }

    // other methods are present but not shown
}
```
a) class CompareByArea implements java.util.Comparator<Rectangle>
{
    public int compare(Rectangle left, Rectangle right)
    {
        int leftArea = left.length * left.height;
        int rightArea = right.length * right.height;

        if (leftArea < rightArea) return -1;
        else if (leftArea == rightArea) return 0;
        else return 1;
    }
}

b) class CompareByArea extends java.util.Comparator<Rectangle>
{
    public int compare(Object left, Object right)
    {
        int leftArea = left.getLength() * left.getHeight();
        int rightArea = right.getLength() * right.getHeight();

        if (leftArea < rightArea) return -1;
        else if (leftArea == rightArea) return 0;
        else return 1;
    }
}

c) class CompareByArea implements java.util.Comparator<Rectangle>
{
    public int compare(Rectangle left, Rectangle right)
    {
        if (left.getArea() < right.getArea()) return -1;
        else if (left.getArea() == right.getArea()) return 0;
        else return 1;
    }
}

d) class CompareByArea extends java.util.Comparator
{
    public int compare(Object left, Object right)
    {
        Rectangle leftRect = (Rectangle) left;
        Rectangle rightRect = (Rectangle) right;

        if (leftRect.getArea() < rightRect.getArea()) return -1;
        else if (leftRect.getArea() == rightRect.getArea()) return 0;
        else return 1;
    }
}
Section J: applying mathematical foundations, algorithmic principles, and computer science theory in the modeling and design of computer-based systems in a way that demonstrates comprehension of the tradeoffs involved in the design choices.

1. **true** false The analysis of a sorting algorithm relates to how steeply the amount of computing resources rises as the size of the data set increases.

2. **true** false Trade-offs between good performance (time complexity) and memory requirements (space complexity) are not unusual when choosing one algorithm over another.

3. **true** false Generally speaking, the binary search gives better performance than height balanced binary search trees, especially in applications where there are numerous insertions and deletions along with the retrievals.
Section K: applying design and development principles in the construction of software systems of varying complexity

1. The adapter pattern converts the API of given class into an API of greater convenience to a client. Given java.util.Stack whose API is abbreviated below.

   public Stack extends Vector
   {
   //Public Instance Methods
   public Object push(Object item);
   public Object pop();
   }

   Our client, a compiler builder, needs a Stack object for evaluating postfix expressions that can push and pop doubles. Show or explain how the adapter pattern may be applied.

   Answer, without exploiting generics and autoboxing and unboxing of JDK 1.5:

   public class Stack  //our adapter in JDK 1.4 and earlier
   {
   java.util.Stack stack = new java.util.Stack();

   public void push(double operand)
   {
   Double doubleObj = new Double(operand);
   stack.push( doubleObj );
   }

   public double pop()
   {
   Object obj = stack.pop();
   Double doubleObj = (Double) obj;
   double operand = doubleObj.doubleValue();
   return operand;
   }
   }

   Answer, using generics and autoboxing and unboxing of JDK 1.5:

   public class Stack  //our adapter in JDK 1.5
   {
   java.util.Stack<Double> stack = new java.util.Stack<Double>();

   public void push(Double operand)
   {
   stack.push( operand );
   }

   public double pop()
   {
   double operand = stack.pop();
   return operand;
   }
   }

   Same driver for either implementation of the new Stack class:

   class Main
   {
   public static void main(String[] args)
   {
   Stack s = new Stack();
   s.push(1.1); s.push(2.2); s.push(3.3);
   System.out.println("first pop: " + s.pop());
   System.out.println("second pop: " + s.pop());
   System.out.println("third pop: " + s.pop());
   }
   }
2. How does aggregation differ from composition?
   a) Aggregation models the **IS-A** relationship between classes, commonly seen as a form of specialization. Composition models the converse, generalization.
   b) In an aggregation association, objects are parts that can come and go such as **Car** objects may come and go from a **Garage** object. In a composition, the objects are integral parts of the whole, such as the **Page** objects making-up a **Book** object.
   c) In UML, the aggregation and composition associations both represented by a diamond of the same shape. The difference is that the diamond for aggregation is filled in.
   d) All object-oriented programming support aggregation but not all offer composition (C++ does; Java does not).

3. Which is **true** of agile software development processes?
   a) **Working software is delivered frequently and is the primary measure of progress.**
   b) **Technical excellence is not one of the developers' concerns.**
   c) **Mechanism are present to discourage requirements changes.**
   d) **Effectiveness is enhanced by insulating the software developers from the customers.**

4. Which is **true** of extreme programming?
   a) It is an agile method that enhances quality with extensive unit testing.
   b) It is an agile method that enhances quality with pair programming.
   c) It is an agile method that emphasizes the collective code ownership.
   d) **All of the above.**
   e) None of the above.
5. Given the following `getSmallest()` method:

```java
class Smallest {
    static int getSmallest(int[] data) {
        int small = Integer.MAX_VALUE;
        if (data.length > 0) {
            for (int i = 0; i < data.length; i++) {
                if (data[i] < small) {
                    small = data[i];
                }
            }
        } else {
            throw new RuntimeException("Empty data array");
        }
        return small;
    }
}
```

The `TestSmallest` class on the next page is:

- a) a properly constructed illustration of JUnit that will output **OK (3 tests)** (because there are three methods with names starting with `test`)

- b) a properly constructed illustration of JUnit that will output **OK (10 tests)** (because there are ten `assert` statements)

- c) a properly constructed illustration of JUnit that will output **OK (11 tests)** (because there are ten `assert` statements and a `fail` statement)

- d) a properly constructed illustration of JUnit, but the output depends on which methods are activated (i.e. the tests selected for use) by the testing driver.
import junit.framework.*;

public class TestSmallest extends TestCase
{
    public TestSmallest(String name) //constructor
    {
        super(name);
    }

    public void testArraysWithNoDuplicateValues()
    {
        int[] a1 = { 9, 8, 7 };
        int[] a2 = { 8, 9, 7 };
        int[] a3 = { 7, 8, 9 };
        int[] a4 = { 8, 7, 9 };
        int[] a5 = { 1, 2, 3, 9, 7, 6, 5, 4 };

        assertEquals(7, Smallest.getSmallest(a1));
        assertEquals(7, Smallest.getSmallest(a2));
        assertEquals(7, Smallest.getSmallest(a3));
        assertEquals(7, Smallest.getSmallest(a4));
        assertEquals(1, Smallest.getSmallest(a5));
    }

    public void testAllValuesTiedForMin()
    {
        int[] a1 = { 3 };
        int[] a2 = { 3, 3 };
        int[] a3 = { 3, 3, 3, 3, 3, 3, 3, 3 };

        assertEquals(3, Smallest.getSmallest(a1));
        assertEquals(3, Smallest.getSmallest(a2));
        assertEquals(3, Smallest.getSmallest(a3));
    }

    public void testForException()
    {
        try
        {
            int[] a1 = { };
            assertEquals(5, Smallest.getSmallest(a1));
            fail("Exception expected: dataset is empty");
        }
        catch (RuntimeException e)
        {
            assertTrue(true);
        }
    }
}
Scoring Instructions
for the
CS Summative Assessment Exam

This test has a total of 41 items distributed among the 11 program outcomes as follows:

<table>
<thead>
<tr>
<th>Program Outcome</th>
<th>Number of items</th>
<th>Maximum Score</th>
</tr>
</thead>
<tbody>
<tr>
<td>A) An ability to apply knowledge of computing and mathematics appropriate to the discipline</td>
<td>5</td>
<td>6</td>
</tr>
<tr>
<td>B) An ability to analyze a problem, and to identify and define the computing requirements appropriate to its solution</td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>C) An ability to design, implement and evaluate a computer-based system, process, component, or program to meet desired needs</td>
<td>5</td>
<td>5</td>
</tr>
<tr>
<td>D) An ability to function effectively on teams to accomplish a common goal.</td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>E) An understanding of professional, ethical and social responsibilities.</td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>F) An ability to communicate effectively with a range of audiences.</td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>G) An ability to analyze the impact of computing on individuals, organizations and society, including ethical, legal, security and global issues</td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>H) Recognition of the need for, and an ability to engage in, continuing professional development</td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>I) An ability to use current techniques, skills, and tools necessary for computing practices</td>
<td>5</td>
<td>6</td>
</tr>
<tr>
<td>J) An ability to apply mathematical foundations, algorithmic principles, and computer science theory in the modeling and design of computer-based systems in a way that demonstrates comprehensions of the tradeoffs involved in the design choices</td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>K) An ability to apply design and development principles in the construction of software systems of varying complexity</td>
<td>5</td>
<td>6</td>
</tr>
</tbody>
</table>

41 items Max score: 44
The mix of question types, accompanied by annotating information, is in the table below:

<table>
<thead>
<tr>
<th>Question type</th>
<th>Number</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>true-false</td>
<td>22 items</td>
<td></td>
</tr>
<tr>
<td>multiple choice</td>
<td>12 items</td>
<td></td>
</tr>
<tr>
<td>short answer</td>
<td>4 items</td>
<td><strong>C.1.</strong> (p. 5) is a postfix expression evaluation.</td>
</tr>
<tr>
<td></td>
<td></td>
<td><strong>I.1.a.i.</strong> (p. 13) requires a line of code (constructor call).</td>
</tr>
<tr>
<td></td>
<td></td>
<td><strong>I.1.a.ii.</strong> (p. 13) requires two lines of code (polymorphic method calls). Each line counts as 1 point.</td>
</tr>
<tr>
<td>code construction</td>
<td>2 items</td>
<td><strong>I.1.b.</strong> (p. 14) requires a concrete class that extends an abstract class and may be used polymorphically. It counts as two points.</td>
</tr>
<tr>
<td></td>
<td></td>
<td><strong>K.1.</strong> (p. 18) requires an implementation or similarly detailed discussion of a <strong>Stack</strong> adapter. It counts as two points.</td>
</tr>
<tr>
<td>mathematical derivation</td>
<td>1 item</td>
<td><strong>A.5.</strong> (p. 3) -- number of edges in a graph when each vertex is directly linked to each other vertex. It counts as two points.</td>
</tr>
</tbody>
</table>

For each of the 38 true-false, multiple choice, or short answer items a correct answer is awarded one (1) point.

Each of the three more elaborate items is worth two (2) points. Full credit (two points) is awarded for an answer that demonstrates a good grasp of what’s involved, even if it is not perfect. Partial credit (one point) signifies understanding at a fair to middling level. The intent of this rough granularity is scoring consistency.

With the grading plan described above, the highest possible score is 44 points. A student receives that when every short-answer question is answered correctly (38 points), and full credit is awarded for the code construction items and the mathematical derivation (6 points).

Our interest, however, is not on the distribution of overall scores but on each of the 11 sections that respectively assesses a program outcome. We are interested in the proportion of students meeting the a priori standard of 70%, in the percentage of students who answered each item correctly, and the class average on each standard.

Scores across the program outcomes can identify program strengths as well as to suggest areas for improvement. Year to year performance can be compared as an effectiveness gauge of assessment-based changes.
The Seidenberg School of Computer Science and Information Systems
Pace University

Technical Report Series

EDITORIAL BOARD

Editor:
Allen Stix, Computer Science, Pace—Westchester

Associate Editors:
Constance A. Knapp, Interim Dean, The Seidenberg School of Computer Science and Information Systems — Pace University
Susan M. Merritt, Computer Science, Pace—Westchester

Members:
Howard S. Blum, Computer Science, Pace—New York
Mary F. Courtney, Computer Science, Pace—Westchester
Nicholas J. De Lillo, Mathematics and Computer Science, Manhattan College
Daniel Farkas, Information Systems, Pace—Westchester
Fred Grossman, Information Systems; Doctor of Professional Studies, Pace—New York and White Plains
Fran Goertzgen Gustavson, Information Systems, Pace—Westchester
Joseph F. Malerba, Computer Science, Pace—Westchester
John S. Mallozzi, Computer Information Sciences, Iona College
John C. Molluzzo, Information Systems, Pace—New York
Pauline Mosley, Technology Systems, Pace—New York
Narayan S. Murthy, Computer Science, Pace—New York
Catherine Ricardo, Computer Information Sciences, Iona College
Judith E. Sullivan, CSIS Alumna, MS in CS from Pace—Westchester
Sylvester Tuohy, Computer Science, Pace—Westchester

The Seidenberg School of Computer Science and Information Systems, through the Technical Report Series, provides members of the community an opportunity to disseminate the results of their research by publishing monographs, working papers, and tutorials. Technical Reports is a place where scholarly striving is respected.

All preprints and recent reprints are requested and accepted. New manuscripts are read by two members of the editorial board; the editor decides upon publication. Statements of policy and mission may be found in issues #29 (April 1990) and #34 (September 1990).

Please direct submissions as well as requests for single copies to:

Allen Stix
The Seidenberg School of Computer Science and Information Systems
Goldstein Academic Center
Pace University
861 Bedford Road
Pleasantville, NY 10570-2799