COEN 79

Object-Oriented Programming and Advanced Data Structures

Assignment #1 - Due: September 29, 2018 – 11:59PM

Name: Jordan Murtiff Date: 9-27-18

- Number of questions: 10
- Points per question: 0.2
- Total: 2 points

1. Please answer the following questions:
   - What is *procedural abstraction*?

Procedural abstraction is a form of information hiding where we only care about what a function does (its precondition and postcondition) not how a function is actually implemented. The procedure of how a function or program is done is hidden away (or abstracted) from us and we care only about what is done and not how it is done.

   - Mention a good approach to detecting invalid data at an early point.

A good approach to detecting invalid data at an early point in a program is to use the assert command in C/C++. The assert command checks whether a certain condition is true, and if the condition is not, then the program is stopped and an error message is shown to the user. We can use this to check if a valid input is used with a function, and if the input is not valid, then the program can stop immediately without executing any more lines of code.

2. What is the time complexity of this algorithm? Present the proof.

```
while ( low <= high )
{
  mid = ( low + high ) / 2;
  if ( target < list[mid] )
     high = mid - 1;
  else if ( target > list[mid] )
     low = mid + 1;
  else break;
}
```

This algorithim is Binary Search, which helps find certain elements through already sorted data types. If we assume that n is the number of items we are searching through, then we know that unless the item we are looking for is in the "middle" of the data type, then we eliminate half of the data set. So if we start at n items, then by the second search we will have n/2 items, and by the third search n/4 items, and so forth.
This can be written as n, n/2^1, n/2^2, n/2^3 and so forth until it reaches a value of 1.

So we then can simplify to n/2^i where "i" is the number of iterations through our data set.

This means we have n/2^i > 1.
Which then can be simplified to n > 2^i (multiply both sides by 2^i) and finally if we take the log of both sides we get that log(n) > i which means that the big O notation for this algorithim is O(log n).

3. What is the time complexity of `fun()`. Present the proof.

```
int fun(int n)
{
  int count = 0;
  for (int i = n; i > 0; i /= 2)
     for (int j = 0; j < i; j++)
        count += 1;
  return count;
}
```

The loops are dependent on each other, therefore we have to use a special method to find the time complexity of fun(). The first thing we will do is calculate what values the two “for” loops will go from: i goes from n to 1 and j goes from n to 1. Now we can write out all the values that variables i and j can have:
i: n + n/2 + n/4 + n/6 + n/8 … + 1
j: n + n/2 + n/4 + n/6 + n/8 … + 1

This means that the outer "for" loop has a big O of O(log n) which is smaller than the inner "for" loop.
If we factor out n from the equation we get n(1/2 + 1/4 + 1/6 + 1/8 ... + 1) and with the infinte series converging to the value of 1 this means that we get n(1 + 1) = n(2) = 2n which simpilies to just O(n).
The time complexity of fun() is O(n).

4. Give a concise formula that gives *the approximate number of digits in a positive integer*. The integer is written in base 10.

Digits = floor(log10(number)) + 1

5. You are at a computer science cocktail party and you meet a student who has just started working with a debugger. With about three or four sentences, explain the basic features of your debugger and how they help you find bugs.

LLDB is the debugger I use when I write programs in C++. It helps me debug by allowing me to set breakpoints throughout my program so that I only run certain lines of code within my program. This is useful in isolating where problems are occurring within my program so that I am able to fix any errors within my code. Additionally, LLDB allows me to print the values of variables between breakpoints so I can ensure that my program variables are reaching their correct values. Finally, LLDB allows me to finally run my program and if any errors come up while running the program, LLDB should tell me the specific line number that causes the error.

6. What are the properties of a *constructor*?

1) A constructor is called automatically whenever a variable of a class (an object) is declared.
2)The name of a constructor must have the same name as the class (although the parameters may be different).
3) A constructor does not have a return type (not even a void return type).

7. What is the difference between a *class* and an *object*? Include an example in your answer.

A class is like a blueprint that defines a kind of data type that has member variables and member functions. An object is an instance of a class and is declared as a variable of a specific class with its own separate member variables (separate from any other objects) and shared member functions (shared between all objects of the same class). There should only be one class, but there can be many objects of a particular class.

```
class animal
{
 public:
  animal();
  string get_name() const;
  void set_weight();
  void set_height();
 private:
  string name;
  double weight;
  double height;
  bool has_eaten;
  }

  And an object of the animal class
            would be:
  animal panda;
```

8. What are the two methods for finding test data that is most likely to cause errors.

The two methods for finding test data that is most likely to cause errors is to:

1) Test boundary values: Although you may not be able to test all possible inputs, ensure that the highest and lowest possible values are tested for your program.

2) Fully exercise code: Make sure every line of code is executed at least once by some test data and also ensure that if part of your code is normally skipped, that the program actually tries to skip the code at least once. This ensures that your program can fully work under all circumstances.

9. What are the three ways we can use items defined in a namespace. Include examples in your answer.

1) using namespace _____ (where the blank is the name of a namespace)
2) using _____::_____ (where the first blank is the name of a namespace and the second blank is a class from that specific namespace)
from a namespace)
3) _____::_____ _____ (where the first blank is the name of a namespace and the second blank is the name of a class and the thrid blank is the name of an object of the class in the second blank)

Examples for all three types are:
1st method: using namespace scu_example;
2nd method: using scu_exmaple::student;
3rd method: scu_example::student joey;

10. Discuss about the output of the following codes:

- Code 1

```
1.  class Test {
2.      int x;
3.  };
4.
5.  int main() {
6.      Test t;
7.      t.x = 20;
8.      getchar();
9.      return 0;
10. }
```

- Code 2

```
1.  struct Test {
2.      int x;
3.  };
4.
5.  int main() {
6.      Test t;
7.      t.x = 20;
8.      return 0;
9.  }
```

Code 1 will not compile. Since Test is a class and not a structure, all of its member variables are private and not public. This means that if Code 1 tries to compile it will come up with an error when we try to assign Test object “t” with a value 20 to its “x” member. For classes in general, all member variables are private and therefore cannot be accessed without special functions.

Code 2 on the other hand will compile. Because Test is a structure and not a class its member variables are all made public. This means that assigning Test object “t” with a value of 20 to its “x” member will work.