COEN 79: Object-Oriented Programming
Homework #1

Tamir Enkhjargal

January 21, 2020

Question #1

What is *procedural abstraction?* Mention an approach for detecting invalid function input data at an early point.

Procedural abstraction is when you pretend to not know how the functions and algorithms are implemented, but you know the conditions. This is done through preconditions and postconditions. You can test if there is an invalid function input through asserting if the input is of the correct type.

Question #2

What is the time complexity of this algorithm? Please present a formal proof of your answer.

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

Assume an array of size n. Then, regardless of however we cut the low, mid, and high, the amount of instructions we will go through is of form $n/2^i$, from $n, n/2, n/4, ..., 2, 1$. Then we can prove that:

$$
\begin{align}
n/2^i &> 1 \\
n &> 2^i \\
log(n) &> log(2^i) \\
log(n) &> i
\end{align}
$$

Therefore, the highest amount of iterations of halving we will have is $log(n)$, and the time complexity of this function is $O(logn)$

Question #3

What is the time complexity of `fun()`. Please prove it.

```
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 variable i starts from n and gets continuously halved each time. j starts from 0 and goes up to i. Therefore the total amount of times `count += 1` is called should be $\sum_{i=0}^{n} n * 2^{-k}$. This is a geometric series, which results in $2n-1$ (convergence of a geometric series). Another method of solving this is to see that the sum is $n + n/2 + n/4 + n/8 + ... + 2 + 1$, and the largest power here is n. Therefore the complexity is $O(n)$.

Question #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
```

Question #5

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

My debugger will be able to tell at best, which line the bug has occurred on, which function it failed at, and in total where it was in the total run of the code (such as if main called the function). Then, it will be able to tell the reason behind the bug, such as is there was an exception thrown (ArrayOutOfBounds, DivideByZero), or if the program was trying to access memory outside its allocated segment. From there, the programmer should be able to know exactly which lines to change.

Question #6

What are the *four* properties of a *constructor*?

- It is always called when an object is instantiated
- Named after the class the constructor is in
- No return type
- Purpose is to set initial values of private variables

Question #7

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

A class is a template that creates objects, and objects are instances of a class, which has actual variables and containers that consumes memory. If `car` is a class, then `Camry` is an object, an instance of a "`car`" and contains information which that car class will have, such as model number, year, milage, functions, etc.

Question #8

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

The first method is the test the boundary values, such as 0, array max, empty queue, full queue, etc. The second method is to fully exercise the code, make sure you are inputting parameters that will run through every single line of code at least once.

Question #9

What are the <u>*three*</u> ways we can use items defined in a *namespace*? Include examples in your answer.

1. Make all of the namespace available (e.g. `using namespace ns_name`)
2. Only using a specific item from the namespace (e.g. `using ns_name::func`)
3. Using them by prefixing using the scope resolution operator in line (e.g. `coen79::car Camry`)

Question #10

Discuss about the output of the following codes:

- Code 1

```
class Test {
    int x;
};

int main() {
    Test t;
    t.x = 20;
    getchar();
    return 0;
}
```

- Code 2

```
struct Test {
    int x;
};

int main() {
    Test t;
    t.x = 20;
    return 0;
}
```

In Code 1, the variable `x` is a private member variable, and the code `t.x = 20` will not work, as it is trying to access a private member variable without a getter method.

In Code 2, the code will work, since structs are defaulted public variables, so you can access a struct's variables without using a method for it.