Queens College, CUNY, Department of Computer Science

Object-Oriented Programming in C++

CSCI 211/611

Summer 2018

Instructor: Dr. Sateesh Mane

due date Friday, July 20, 2018, 11.59 pm

*** UNDER REVIEW ***
*** DO NOT DO ***

Homework: Classes: functions and methods

- **Experience with other classes has demonstrated that in many cases the source of difficulty is not the mathematics or the programming.**
- **The source of difficulty is the English (understanding the text).**
- **If you do not understand the words in the lectures or homework, THEN ASK.**
- **If you do not understand the concepts in the lectures or homework, THEN ASK.**
- **Send me an email, explain what you do not understand.**
- **Do not just keep quiet and then produce nonsense in exams.**

- **Consult your lab instructor for assistance.**
- **You may also contact me directly, but I cannot promise a prompt response.**
- Please submit your inquiry via email, as a file attachment, to `Sateesh.Mane@qc.cuny.edu`.
- Please submit one zip archive with all your files in it.
  1. The zip archive should have either of the names (CS211 or CS611):

     `StudentId_first_last_CS211_hw_classes1.zip`
     `StudentId_first_last_CS611_hw_classes1.zip`
  2. The archive should contain one "text file" named "hw_classes1.[txt/docx/pdf]" and one cpp file per question named "Q1.cpp" and "Q2.cpp" etc.
  3. Note that not all questions may require a cpp file.

General information

- You should include the following header files, to run the programs below.

  ```
  #include <iostream>
  #include <iomanip>
  #include <string>
  #include <cmath>
  ```

- If you require additional header files to do your work, feel free to include them.
- **Include the list of all header files you use, in your solution for each question.**
- The questions below do not require complicated mathematical calculations.
- If for any reason you require help with mathematical calculations, **ask the lab instructor or the lecturer.**

Q1 Classes: functions and methods

- **Write a class `PowMult` to calculate the power x^n and product $x * n$ of a double x and integer n.**

- The class has one private data member x.

```
class PowMult
{
public:
  // to do

private:
  double x;
};
```

- We shall write additional class methods, to be described below.

- We shall employ two algorithms to calculate the power x^n, to be explained below.

- **We shall ignore problems of overflow and underflow.**

Q2 Default constructor

- **Write a default constructor for the class.**

```
PowMult() // fill in the rest
```

- Set $x = 0$ in the constructor. It should be obvious how to do this.

Q3 Accessor and mutator

- **Write accessor and mutators methods to get and set the value of x.**

```
double get() const          // fill in the rest
void set(double d)          // fill in the rest
```

- There is only one data member, so we name them simply "`get`" and "`set`" respectively.
- They are both public.
- It should be obvious how to write both methods.
- The accessor method `get()` returns the value of x.
  1. The return type is `double`.
  2. The method is `const`.
- The mutator method `set(double d)` sets the value $x = d$.
  1. The return type is `void`.
  2. The method is not `const`.

Q4 Methods: `power(...)` and `mult(...)`

- **Write a method to calculate the power x^n.**

```
double power(int n);
```

- The method is public.
- The return type is `double`.
- The return value is `pow(x,n)`.
- **Question: can the method `power` be tagged as `const`?**
- **Write a method to calculate the product $x * n$.**

```
double mult(int n);
```

- The method is public.
- The return type is `double`.
- The return value is `x*n`.
- **Question: can the method `mult` be tagged as `const`?**

Q5 Class declaration

- **Your overall class declaration should look like the following.**

```
class PowMult
{
public:
  PowMult();

  double get() const;
  void set(double d);

  double power(int n);            // const or not ??
  double mult(int n);             // const or not ??

private:
  double x;
};
```

Q6 Functions

- **Write two functions as follows to use your code.**

  ```
  void printLoop(int n, PowMult &p);
  void printLoop(int n, PowMult *q);
  ```

- **In the second function, if q is NULL then return immediately.**
- In both functions, if $n < 0$ then set $n = -n$. Then run a loop for $i = 1, \ldots, n$.
- In the loop, print the values of i, `power(i)` and `mult(i)`.
- **Use the dot and arrow operators correctly in each function.**

  ```
  void printLoop(int n, PowMult &p)
  {
    if (n < 0) n = -n;
    for (int i = 1; i <= n; ++i) {
      // print value of i, power(i),  mult(i) for reference p
    }
  }

  void printLoop(int n, PowMult *q)
  {
    //  *** test if q is NULL ***
    if (n < 0) n = -n;
    for (int i = 1; i <= n; ++i) {
      // print value of i, power(i),  mult(i) for pointer q
    }
  }
  ```

Q7 Example main program

Q7.1 Single objects

- **Write a main program to test your code.**
- Here are some variables.

```
int n = 4;
double x = 2.0;
PowMult p;
PowMult *q = new PowMult;
```

- **Call `set(x)` in the correct way for p and q.**
- **Execute the following function calls.**

```
printLoop(n, p);
printLoop(n, &p);                // &p is pointer
printLoop(n, q);
```

- **Remember to release the memory for q correctly.**

Q7.2 Arrays of objects

- **Write the following code and run it.**
- Make sure you can explain all the lines of the code.
- **Which version of `printLoop` is called?**

```
PowMult a[2];                // array of objects
a[0].set(x);               // syntax for array element and dot operator
a[1].set(sqrt(x));
printLoop(n, a[0]);
printLoop(n, a[1]);

PowMult *b = new PowMult[2];   // dynamically allocated array of objects
b[0].set(x);
b[1].set(sqrt(x));
printLoop(n, b[0]);
printLoop(n, b[1]);
```

- **Remember to release the memory for b correctly.**