EEM 480 Homework 2

Due to October 24 2020

Write a JAVA program using object orienting design technique of name Absurd. In this program you are required to design a class with methods and properties. The class structure should like to be given below:

```
class Absurd {
public:
	//constructor
	Absurd(//You have to design the parameters)
	{}
	//check whether the string is palindrome and informs the user
	bool Palindrome()
	{}
	//user can enter his/her string
	GetString()
	{}
	//delete the k'th letter in string x
	Delete(k)
	{}
	// reverse the string and put the new reversed form to screen using
	   PrintString function
	OutReverse
	{}
	// reverse the string in words and put the new reversed form to
	   screen using PrintString function
	WordReverse
	{}
	//Insert x string after the k'th point
	Insert(k,x)
	{}
	//change every letter in the string to its capitilized form
	MakeCapitilize()
	{}
	//put the string into the output stream out
	Output()
	{}
	//read the string from the file
	InFile(String filename)
	{}
	//write the string to file
	OutFile(String filename)
	{}

private:
	PrintString(//You have to design the parameters)
	{}
	MyString // Design a instance to keep the string
};
```

Here the user can use the program by giving same commans as below:

The commands are :
s or S The user can enter the string

d/D *n* the *n*'th character is deleted from the string and the result is shown
p/P Program check whetheer the sring is palindrome or not
r/R The string will be reversed and result is shown
i/I n <String> The <String> will be inserted after *n*'th position of the string and the result is shown
m/M The string will be capitilized and result is shown
o/O output string to screen
t/T<Pathname\FileName> The string is read from the file
w/W<Pathname\FileName> The string is written to the file
f/F Program reverse the words and result is shown
x/X Terminates the program

Example

```
>>s Here is the String
>>Here is the String
>>d 4
>>Her is the String
>>p
>> The String is not a palindrome
>>r
>>gnirtS eht si reH
>> i 5 e
>>gnirteS eht si reH
>>o
>> gnirteS eht si reH
>>m
>>GNIRTES EHT SI REH
>>r
>>HER IS THE SETRING
>>f
>>SETRING THE IS HER
>>t c:\Myfile.txt
>> The file has been opened and the string has been read
>>w c:\Yourfile.txt
>> The file has been opened and the string has been written
>>x
C:\>The Program has terminated
```

Rules for HW Submission

. You must write your HW in NetBeans environment.
. You must write a report with name "**Report_HW2.pdf**" explaining your HW (purpose, how did you solve it, algorithm etc.) and what you the environment you used (NetBeans, for example). The person who read your report can easily use the class you have written.
. Discuss the result you have obtained.
. Submission should be in the form of a zip/rar. When extracted, the result should be a single folder with the name "HW2".
. Do not forget to put your report into the zip/rar file.
. The name of your project will be "**Name_Surame_HW2**. e.g. *Lutfullah_Arici_HW2*. If you do not obey the rule I will not grade your homework.
. You must bundle your whole project folder into your HW2.zip file.
. If I extract your project file, then import to my environment and if it doesn't work, you will be graded on 30 not 80. (Double check. It saves life)
. Do HW by yourself. Be honest.

Grading Criteria:
Implement a class that has fields for all the data contained in one data set (i.e. an integer, a floating-point number, a character, and a string), and member functions that perform the required computation and output. Data fields of your class should be declared as “private” members of the class, so you will need to implement get() and set() functions to access them. Member functions that perform computation and output should be “public” members of the class. Your program should then be modified so that it uses this class to store the data in a data set and to generate the required output.

A fully working program is worth 80 points. A useful and descriptive report file is worth 10 points. Useful comments throughout your program are worth 15 points. Clear, easy to read syntax (i.e. following coding style guidelines) is worth 10 points.

EEM 480
Coding Guidelines
What follows is a list detailing standards to be followed whenever you are programming for EEM 480 - Variable and function names are to be descriptive of what the variable represents, or what the function does (i.e. “int numDataSets” instead of “int n”). - The first word in the name of a variable or function is to be written in all lowercase letters. The first letter of each subsequent word is to be capitalized. Special characters (underscores and the like) should not be used except for in special cases (i.e. “int someReallyLongVariableName”). - Names of classes follow the same rule, with the exception that the first letter of the first word is also capitalized (i.e. “class MyClass”). - Proper spacing and indentation should be used (see examples):

```
//GOOD
for(int loopCount = 0; loopCount < 10; loopCount++)
{
        if (loopCount >= 4) //countdown from 5 to 0
        {
                System.out.print( 9 - loopCount );
        }
}
//BAD
for(int loopCount=0;loopCount<10;loopCount++)
{
if(loopCount>=4) //countdown from 5 to 0
{
System.out.print( 9 - loopCount );
}
}
```
- Class data members should be declared as “private” members of the class unless there is a compelling reason to do otherwise (and occasionally, there will be). - Class function members should generally be declared as “public” members of the class (though there will be more frequent exceptions to this rule than to the above rule).
- Global variable declarations should be avoided except in rare instances (for example, having a global variable that specifies whether “debug” mode is on or off).
- Generally speaking, a function that does computation should produce no output (except possibly for debug output), and a function that generates output should perform little to no relevant computation. Instead, have the function that performs computation return data (or a data structure) that can then be passed to the function that generates your output. The purpose of this is to keep your modules as specific and reusable as possible.
- Every function in a program should include comments detailing, at a minimum, what the function does, what the parameters are, and what the return value is.
- In instances where a descriptive enough variable name cannot be found while still using a reasonable amount of characters, additional comments are to be used to explain what the variable is used for.
- Particularly confusing pieces of code are to be accompanied by additional comments explaining what the code is doing (if it’s confusing to you, as the person who wrote it, think how much worse it’s going to be for a complete stranger trying to just look at it and understand what’s going on).