**Object Oriented Programming**

**Deadline: Monday, November 21, 2022, 12:10AM**

**Submission must be on Portal**

|  |  |
| --- | --- |
| **Assignment No. 1** | |
| **Topic** | Classes, constructors, destructor, setters and getters,shallow vs deep copy, memory leakage and dangling references issues. |
| **Objective/**  **Outcome** | Making students familiarize with Classes, constructors, destructor, setters and getters,shallow vs deep copy, memory leakage and dangling references issues. |

**Instructions:**

* Indent your code.
* Comment your code.
* Use meaningful variable names.
* Plan your code carefully on a piece of paper before you implement it.
* The name of the program should be the same as the task name. i.e. the first program should be Task\_1.cpp
* Do all the tasks in multiple files (separate interface and implementation).
* All the tasks must be in a single folder (having .h and .cpp files)
* void main() is not allowed. Use int main()
* You are not allowed to use system("pause")
* You are not allowed to use any built-in functions
* Implement it in multiple files. Every task will contain three respective files Class.h Class.cpp and task.cpp (main().cpp)

**You are required to follow the naming conventions as follows:**

* **Variables:** firstName; (no underscores allowed)
* **Function:** getName(); (no underscores allowed)
* **ClassName:** BankAccount (no underscores allowed)

**Task 1:**

Define a class **MyNum** having the following private attributes

* A single integer variable.

**Now do the following operations on above-mentioned class MyNum:**

* Write **parameterized constructor with default arguments.**
* Write separate setters for each attribute to set value.
* Write separate getters for each attribute to get value.
* A function that can convert the number **stored in MyNum** into positiveNumber.
* A function that can convert the number **stored in MyNum** into negativeNumber.

**Now write a driver program (main()) to test the following functionalities on MyNum class on your own.**

* Make a single MyNum object by taking data from user.
* Print a single MyNum object data on screen.
* Make N MyNum objects by taking data from the user, where N is asked from the user.
* Print N MyNum objects data on screen.
* Sort the N MyNum objects by values.
* Print the Sorted MyNum objects on screen.

**Task 2:**

Define a class **MyChar** having the following private attributes

* A single char variable.

**Now do the following operations on above mentioned class MyChar:**

* Write **parameterized constructor with default arguments.**
* Write separate setters for each attribute to set value.
* Write separate getters for each attribute to get value.
* A function that can convert the number **stored in MyChar** into upperCase.
* A function that can convert the number **stored in MyNum** into lowerCase.

**Now write a driver program (main()) to test the following functionalities on MyChar class on your own.**

1. Make a single MyCharobject by taking data from user.
2. Print a single MyCharobject data on screen.
3. Make N MyCharobjects by taking data from the user, where N is asked from the user.
4. Print N MyCharobjects data on screen.
5. Sort the N MyCharobjects by ascii.
6. Print the Sorted MyCharobjects on screen.

**Task 3:**

Make a class **Name** and check all your functions at least once in the main program. The description of class name is given in the header file below:

class Name{

private:

char\* firstName;

char\* lastName;

public:

Name(char\* first=nullptr ,char\* last=nullptr);// parameterized constructor with default values

Name(const Name&); //copy constructor

~Name();//destructor

void copyName(Name&); //this is not a copy constructor, I want this function to copy the contents of one name(firstName and lastName both) to another name.

void camelCase(); // make first letter capital of both attributes

void toLower(); //convert name to lower case alphabets

void toUpper(); //convert name into upper case alphabets

int nameLength(); // both first and last (excluding space)

void swapNames(); // firstName becomes lastName and vice versa

void display(); //prints name(firstName and lastName with space in between)

char\* fullName(); //concatenate both attributes and return full name with a space in between both

bool isValidName();// name should contain only alphabets - no special characters or digits

**////provide setters/getters for Name class. Remember: Setters with no memory leakage.**

**// provide other relevant methods you want like your own strLength and strCopy function in a separate file known as helperFunctions.h and helperFunctions.cpp etc;**

};

You have to provide a **nameCompare** function which is not the member of class name.(can be written in source.cpp in which you have written main function).

**int nameCompare(Name name1, Name name2)** :

This function takes two names as arguments and compare these two names, first you have to compare lastName(lexicographically) and then firstName(lexicographically) **Hint:** Compare only if the result of matching Last Names is returned as **matched**.

 **nameCompare**() compares the **two names lexicographically** means it starts comparing character by character starting from the first character until the characters in both names are equal or a NULL character is encountered.

 If the first character in both names is equal, then this function will check the second character, if this is also equal then it will check the third, and so on

 This process will be continued until a character in either name is NULL or the characters are unequal.

This function can return **three different integer values** based on the comparison:

* **Zero ( 0 )**: A value equal to zero when both names are found to be identical. That is, all of the characters in both names are the same.
* **Greater than zero ( >0 )**: A value greater than zero is returned when the first not matching character in name1 have the greater ASCII value than the corresponding character in name2 or we can also say If character in name1 is lexicographically **after** the character of name2.

* **Less than Zero (<0 )**: A value less than zero is returned when the first not matching character in name1 has a lesser ASCII value than the corresponding character in the name. If a character in name1 is lexicographically **before** the character of name2.

**Important point:** When the names are not the same, you will find that the value returned by the nameCompare() function is the difference between the ASCII values of the first unmatched character in name1 and name2 in both cases.