Assignment-4

What to Hand In

Save your .cpp files for Problems 2, 3, 4, and 5 as FirstName_LastName_ProbX_Assignment4.cpp. Please submit (upload) your source code to Canvas. **Please provide snapshots of all your results after running your code.**

Problem-1 (10 points):

1) The statement **delete p**; (Choose correct answer)

   a) deallocates the variable pointer p.
   b) deallocates the dynamic variable that is pointed to by p.

2) Write a piece of code for the following statement:

   Write a statement that dynamically allocates an array of 50 components of type int, and p contains the base address of the array.

int* p = new int[50];

3) Suppose that you have the declaration int *numPtr;. What is the difference between the expressions: *numPtr and &numPtr?

*numPtr is the value that is being pointed to, while &numPtr is the address where that value is being stored.

4) What is wrong with the following C++ code?

```
double *firstPtr = new double; //Line 1
double *nextPtr = new double; //Line 2
*firstPtr = 62; //Line 3
nextPtr = firstPtr; //Line 4
delete firstPtr; //Line 5
delete nextPtr; //Line 6
firstPtr = new double; //Line 7
*firstPtr = 28; //Line 8
cout << *firstPtr << " " << *nextPtr << endl; //Line 9
```

The pointer is being deleted twice. Since nextPtr = firstPtr, when you delete firstPtr and delete nextPtr, you already deleted the first one so it tries to delete itself again.

5) Consider the following

statement: `int *num;`

a. Write the C++ statement that dynamically creates an array of 10 components of type int and num containing the base address of the array.

```
int *num = new int[10];
```

b. Write a C++ code that inputs data into the array num from the standard input device (Hint: use cin with a loop to fill out the array).

```
for (int i = 0; i > 10; i++)
{
        cin >> num[i];
}
```

c. Write a C++ statement that deallocates the memory space of array to which num points.

```
delete[] num;
```

6) Explain the difference between a shallow copy and a deep copy of data.

A shallow copy means that both the original variable and the copy share the same data, so changes to one will affect the other. A deep copy creates a separate copy of the data, so changes to the copied variable do not affect the original.

Example: Car car1(“BMW”); Car car2 = car1; car2.setModel(“Ford”);

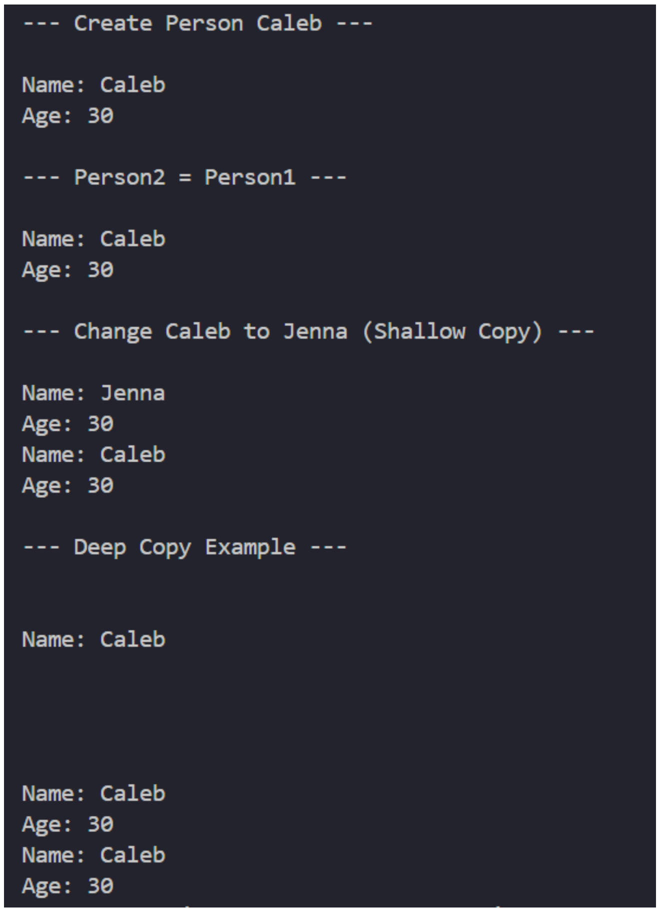
car1 would also change to Ford if it was a shallow copy, while car1 would stay as BMW if it was a deep copy.

Problem-2 (5 points):

The program aims to illustrate the concepts of shallow copy and deep copy in C++ using a **Person** class.

- Create a class named **Person** to represent an individual with attributes **name** and **age**.
- Please use private int age and private char pointer for the name. Please do not use string or char array.
- The class should include constructors, a destructor, and a display method.
- Implement a constructor that initializes a **Person** object with a name and an age. Inside this constructor, allocate memory for the name and copy the provided name into it.
- Implement copy constructor for deep copy, which allocates new memory for the name and copies its content.
- Define a destructor that frees the memory allocated for the **name** attribute.
- Implement a **display** method to display the name and age of the **Person** object.
- In the **main** function, create instances of the **Person** class.
- Create a **Person** object using a shallow copy and display the information (p2 = p1;).
- Create a **Person** object using a deep copy method and display the information.
- Modify the name attribute of one of the **Person** objects created earlier to observe the impact of shallow and deep copy.

Finally, display the information of all **Person** objects to demonstrate the effects of shallow and deep copy.


```
--- Create Person Caleb ---

Name: Caleb
Age: 30

--- Person2 = Person1 ---

Name: Caleb
Age: 30

--- Change Caleb to Jenna (Shallow Copy) ---

Name: Jenna
Age: 30
Name: Caleb
Age: 30

--- Deep Copy Example ---


Name: Caleb



Name: Caleb
Age: 30
Name: Caleb
Age: 30
```

Problem-3 (10 points):

Design a class named MySet which can hold integers from 0 to a defined range. The set is represented as a dynamic array of Boolean values. Element with the index j is true if integer j is in the set. The element with the index k is false if integer k is not in the set. The class contains:

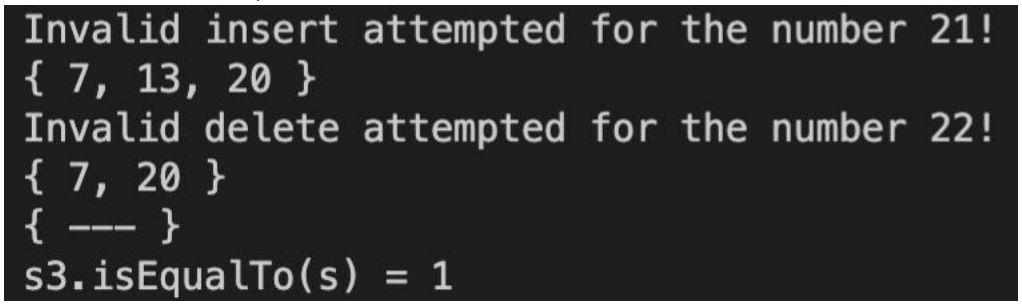
- Private unsigned int data members range that represents the biggest possible integer in a set.
- Private size that represents a number of elements in a set.
- Private pointer to bool named set.
- A user-defined constructor MySet(unsigned) that creates a MySet object with dynamic array with the range of elements from 0 to the given number inclusively and initializes the set of elements to false.
- A copy constructor MySet(const MySet &) that creates a set with a given MySet object.
- A destructor that destroys dynamically allocated array.
- A function void insertElement(int k) that places a new integer k into a set by setting a[k] to true. Function should check if the given number is in the range of a set numbers and print a warning message if the given number is out of the range.
- A function void deleteElement(int m) that deletes integer m by setting a[m] to false. Function should check if the given number is in the range of a set numbers and print a warning message if the given number is out of the range.

- A constant function void printSet() that prints a set as a list of coma-separated numbers. The function prints only those elements that are present in the set. It also prints **{ --- }** for an empty set. You should implement the required output format (see the sample run).
- A constant function bool isEqualTo(const MySet&) that determines whether two sets are equal.

Write a test program (testMySet.cpp) that does the following:

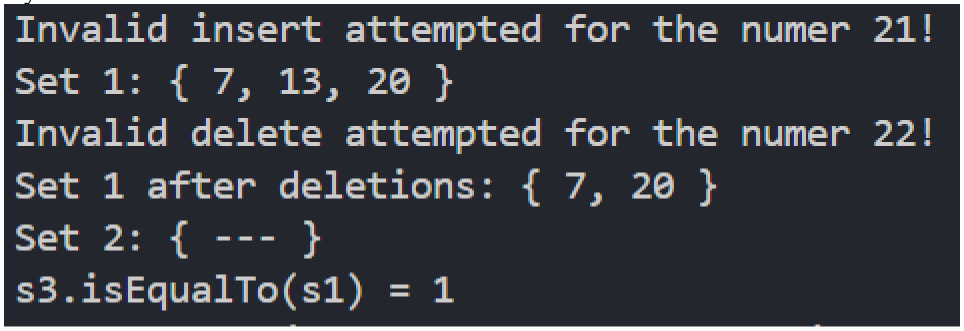
1. Creates a first MySet object with the range of numbers [0; 20].
2. Inserts numbers 13, 7, 20, 21.
3. Calls printSet function from the created object.
4. Deletes numbers 22 and 13.
5. Calls printSet function from the created object.
6. Creates a second MySet object with the range of numbers [0; 100].
7. Calls printSet function from the second object.
8. Creates a third MySet object with the copy constructor with the first object as an argument.
9. Checks is the third set equals to the first set by calling isEqualTo function from one of the objects and prints the result.

Here is a sample run:



```
Invalid insert attempted for the number 21!
{ 7, 13, 20 }
Invalid delete attempted for the number 22!
{ 7, 20 }
{ --- }
s3.isEqualTo(s) = 1
```

My run:



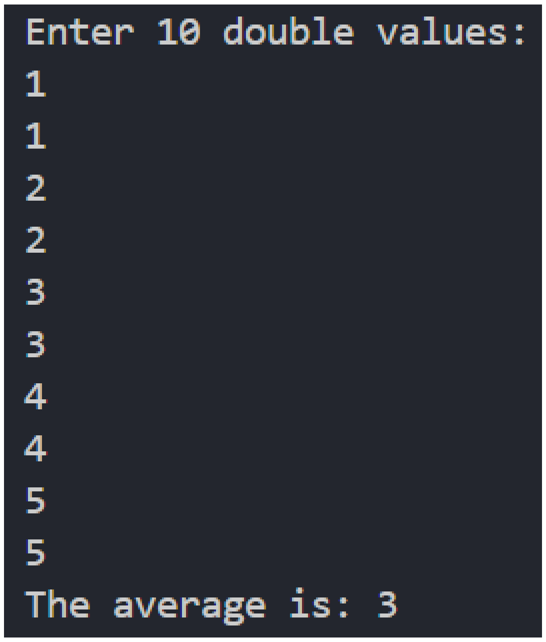
```
Invalid insert attempted for the numer 21!
Set 1: { 7, 13, 20 }
Invalid delete attempted for the numer 22!
Set 1 after deletions: { 7, 20 }
Set 2: { --- }
s3.isEqualTo(s1) = 1
```

Problem-4 (5 points):

Write two overloaded functions that return the average of an array with the following headers:

int average (const int* array, int size);
double average (const double* array, int size);

Write a test program that prompts the user to enter ten double values, invokes this function, and displays the average value.



```
Enter 10 double values:
1
1
2
2
3
3
4
4
5
5
The average is: 3
```

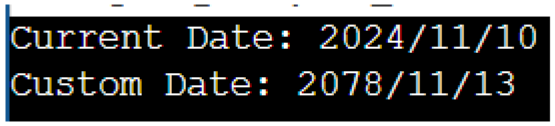
Problem-5 (10 points):

Design a class named **MyDate**. The class contains:

- The data fields **year, month**, and **day** that represent a date. **month** is 0-based, i.e., **0** is for January.
- A no-arg constructor that creates a **MyDate** object for the current date.
- A constructor that constructs a **MyDate** object with a specified elapsed time since midnight, January 1, 1970, in seconds.
- A constructor that constructs a **MyDate** object with the specified year, month, and day.
- Three constant **get** functions for the data fields **year, month**, and **day**, respectively.
- Three **set** functions for the data fields **year, month**, and **day**, respectively.
- A function named **setDate(long elapsedTime)** that sets a new date for the object using the elapsed time.

Implement the class. Write a test program that creates two **MyDate** objects (using **MyDate()** and **MyDate(3435555513)**) and displays their year, month and day.

Hint: The first two constructors will extract the year, month, and day from the elapsed time. For example, if the elapsed time is **561555550** seconds, the year is **1987**, the month is **9**, and the day is **18.**)



```
Current Date: 2024/11/10
Custom Date: 2078/11/13
```

Rubric for Implementation Problems

0%	25%	50%	75%	100%
Source code files were not provided. Problem solution was not submitted.	Significant assignment requirements were ignored or violated. Program doesn’t compile.	The output of the program was not shown. Lack of comments. Pour code readability (inconsistent indentation, variable naming, general organization)	Choosing a poorly approach to solve a problem, for example, solving a problem with hard coding instead of using a loop. Minor details of the program specifications were violated.	Program works correctly and meets the requirements of the assignment. Code is clean, well-organized, and well commented.