CMPT 135: Lab Work Week 7

1. The following program has a runtime error. What is the cause of the error? Fix it and get it to run correctly.

```
int main()
{
    srand(time(0));

    //Construct a default vector object to store integers
    vector<int> a;

    //Put five integer values in the vector
    cout << "Populating the vector" << endl;
    for (int i = 0; i < 5; i++)
            a[i] = rand() % 20;

    //Print the elements of the vector
    cout << "Printing the vector elements" << endl;
    for (vector<int>::iterator it = a.begin(); it < a.end(); it++)
            cout << *it << endl;

    system("Pause");
    return 0;
}
```

2. The following program has a runtime error. What is the cause of the error? Fix it and get it to run correctly.

```
int main()
{
    srand(time(0));

    //Construct a default vector object to store integers
    vector<int> a;

    //Put five integer values in the vector
    cout << "Populating the vector" << endl;
    for (int i = 0; i < 5; i++)
            *(a.begin() + i) = rand() % 20;

    //Print the elements of the vector
    cout << "Printing the vector elements" << endl;
    for (vector<int>::iterator it = a.begin(); it < a.end(); it++)
            cout << *it << endl;

    system("Pause");
    return 0;
}
```

3. Write a function named pop_front that takes a vector of type int and removes the first element in the vector.

4. Write a binary operator function **+** that adds two vectors of double data type. Your operator should not modify the operands and it must return a new vector of double data type that contains the elements of the left hand side operand followed by the elements of the right hand side operand.

5. Write a binary operator function ***** that multiplies a vector of int data type left hand side operand with an integer right hand side operand. Define multiplication as duplicating the vector as many as the integer value times. For example, given the vector a = [1, 2, 3], then a*5 should give [1,2,3,1,2,3,1,2,3,1,2,3,1,2,3]. Your function should not modify its operands but rather return a new vector result.

Think

- What should a*0 return?
- What should an empty vector * some integer value return?
- What will happen if the right hand side operand is negative integer?

6. Write a function named swap that takes two vectors of char data type as arguments and that swaps the arguments.
7. Write a binary operator function **-** that will subtract the right hand side vector from the left hand side vector. Assume int data type vectors. Your operator should not modify the operands and it must return a new vector that contains the elements of the left hand side operand but are not found on the right hand side operand.
8. Write a function named **`isDistinct`** that takes a vector of int data type and returns true if the vector contains distinct elements; otherwise returns false. Note that the elements of a vector are said to be distinct if each element of the array is different from every other element.
9. Write a function named **`distinctElements`** that takes a vector of int data type and returns a new vector containing the distinct elements of the argument. Your function should not modify the argument. For example if the vector argument contains the elements [2, 4, 2, 6, 3, 6] then your function must return a new vector containing [2, 4, 6, 3].
10. Write a function named **`removeDuplicates`** that takes a vector of int data type and modifies the argument so that only the distinct elements of the vector are kept while any duplicate is removed. For example if the vector argument contains the elements [2, 4, 2, 6, 3, 6] then your function must modify the argument vector so that it will have only the elements [2. 4, 6, 3].
11. Write a C++ program that declares a default vector of int, pushes back five random integers in the range [0, 5] onto the vector, and then duplicates each element of the vector as many times as the value of the element itself. For example if the vector is initially [2, 1, 4, 0, 5] then at the end the element 2 should be duplicated twice, 1 should be duplicated once, 4 should be duplicated 4 times, 0 should be duplicated 0 times and 5 should be duplicated 5 times; which means we should get the vector [2, 2, 1, 4, 4, 4, 4, 5, 5, 5, 5, 5] at the end.
12. Write a function named **`insertIncreasing`** that takes a vector of integers whose elements are sorted in increasing order and an integer value as arguments and inserts the integer argument in the vector such that after the insertion operation, the elements of the modified vector are also in increasing order. Here is a very simple test code for you. Insert as many integers as possible for rigorous test.

```
#include <iostream>
#include <vector>
using namespace std;

void printVector(const vector<int> &x)
{
	for (int i = 0; i < x.size(); i++)
		cout << x[i] << "  ";
	cout << endl;
}
void insertIncreasing(vector<int> &a, const int x)
{
	//Fill your code here
}
int main()
{
	vector<int> a;
	for (int i = 0; i < 5; i++)
	{
		int num = rand() % 11 - 5;
		cout << "Insering " << num << endl;
		insertIncreasing(a, num);
		printVector(a);
	}
	system("Pause");
	return 0;
}
```

Sample output should be as follows:

```
Inserting 3
3
Inserting 8
3   8
Inserting -2
-2   3   8
Inserting 6
-2   3   6   8
Inserting 0
-2   0   3   6   8
```

13. Write a function named **`insertGrouped`** that takes a vector of integers whose elements are grouped (i.e. its even integer elements are grouped together and its odd integer elements are also grouped together) and an integer value as arguments and inserts the integer argument in the vector such that after the insertion operation, the elements of the modified vector are also grouped. Use a similar test code as Q12 above.

14. Write a function named **`insertGroupedIncreasing`** that takes a vector of integers whose elements are grouped into even and odd groups and within each group they are sorted in increasing order and an integer value as arguments and inserts the integer argument in the vector such that such that after the insertion operation, the elements of the modified vector are also grouped into even and odd groups and within each group elements are sorted in increasing order. Use a similar test code as Q13 above.

15. Write a function named **`reOrderElements`** that takes a vector of integers and that re-orders the elements of the vector so that the even integer values (elements) of the vector are grouped together and the odd integer values (elements) of the vector are also grouped together.

For example given the vector v = [3, -4, 0, 5, 2, 1, 5, 6], the function call

```
reOrderElements (v);
```

modifies the vector so that it is modified to [5, 1, 5, 3, -4, 0, 2, 6].

Please note that the order of the elements within each group is not important. What is needed is the even integer values to be grouped together and the odd integer values to be grouped together as well.

Hint: Use your `insertGrouped` function defined above.

16. Consider the following inventory list that contains the information of vehicles available in a vehicle dealer shop. The first column in the inventory lists the make of vehicles, the second column lists the year of vehicles and the third column lists the horse power of vehicles.

```
Nissan          2012            750.50
Toyota          2017            650.00
Jeep            2011            650.00
Hyundai         2018            500.00
Hyundai         2008            700.50
Mazda           2014            500.00
Kia             2010            750.50
Kia             2012            700.00
Jeep            2012            750.00
Toyota          2013            600.50
Kia             2018            700.00
Jeep            2008            750.00
BMW             2009            600.50
Jeep            2017            550.00
Honda           2013            500.00
Mazda           2012            700.00
BMW             2009            750.00
Jeep            2005            650.00
Hyundai         2010            750.50
Toyota          2008            750.00

⋮
```

In order to store a vehicle information, let us declare a C++ struct as follows:

```
struct Vehicle
{
        string make;
        int year;
        double hp; //horse power of the vehicle
};
```

Write a C++ function named **`readVehicleInventory`** that takes a string argument which is the full path of an input text file containing an inventory similar to the sample shown above, reads all the vehicle information in the input file to a vector of **`Vehicle`** structs and finally returns the vector of **`Vehicle`** structs. Observe that we do not know how many vehicle objects are listed in the input file.

17. Write a function named **`reOrderObjects`** that takes a vector of **`Vehicle`** structs and that re-orders the elements of the vector so that the same make elements of the vector are grouped together in the vector.

18. Write a function named **`mostPowerful`** that takes a vector of **`Vehicle`** structs and a **`string`** representing the make of a vehicle arguments and that returns the element of the vector with the highest power among all elements of the same make as the string argument.