The University of Central Punjab

Faculty of Information Technology

Data Structures and Algorithms
Fall 2022

Lab 03	
Topic	• Understanding Classes • Working on Three Different files • Abstract Classes • Templates • Arrays • Big-O time complexity
Objective	The basic purpose of this lab is to revise some preliminary concepts of C++ that have been covered in the course of Introduction to Computing and Programming Fundamentals and Object-Oriented Programming.

Instructions:

- Indent your code.
- Comment your code.
- Use meaningful variable names.
- Plan your code carefully on a piece of paper before you implement it.
- Name of the program should be the same as the task name. i.e. the first program should be Task_1.cpp
- **void main() is not allowed. Use int main()**
- **You have to work in multiple files. i.e separate .h and .cpp files**
- **You are not allowed to use system("pause")**
- **You are not allowed to use any built-in functions**
- **You are required to follow the naming conventions as follow:**
    - **Variables:** firstName; (no underscores allowed)
    - **Function:** getName(); (no underscores allowed)
    - **ClassName:** BankAccount (no underscores allowed)

Students are required to complete the following tasks in lab timings.

Task 1

Create a C++ generic abstract class named as **List**, with the following:

Attributes:

1. Type * arr;
2. int maxSize;
3. int currentSize;

Functions:

virtual Type removeElementFromSpecificPositionList() = 0;

- Should remove the element from the specific position of the **List**

virtual void firstRepeatingElement(Type) = 0;

- Should return the first repeating element in a **List**

virtual void firstNonRepeatingElement(Type) = 0;

- Should return the first Non repating element in a **List**

virtual Type findPairs(Type) = 0;

- Should return all paris whose sum is equal to target value from the **List**
- **Target Sum = 7**

2	**5**	**3**	**4**	**-1**

- **Output Pairs are : (2,5), (3,4).**

- Write parameterized constructor with default arguments for the above class.
- Write Destructor for the above class.

Task 2

Create a menu based program for the following functions, using the class made in task 1, make a class named as **MyList**, having following additional functionalities:

countFrequencyOfEachElement() : List which may contain duplicates, print all elements and their frequencies.

-4	-3	1	-2	1	1	-3	-5

Counting Frequancy of Each Element :

Element -4 its frequency is 1

Element -3 its frequency is 2

Element 1 its frequency is 3

void leftRotate(arr[], size) : Rotate List in left mode.

Input Array:

1	2	3	4	5

leftRotate = 2

Output Array After Left Rotation:

3	4	5	1	2

void leftRotate(arr[], size) : Rotate List in right mode.

Input Array:

1	2	3	4	5

rightRotate = 2

Output Array After Right Rotation:

4	5	1	2	3

- Write parameterized constructor with default arguments for the above class..
- Write Destructor for the above class.

Task 3 (Class Participation Task)

Compute the complexity of the following codes, according to their line by line execution. Finally compute their big(O) complexity.

Algorithm Example:

1. int Max = 0 //assume S[N] is filled with +ve numbers

```
2.  For (int I = 0;I<N;I++)
3.  {
4.      If (S[I] > Max)
5.          Max = S[I]
6.  }
7.  cout<< Max
```

Solution:

Instruction	No. of Times Executed	Cost
1	1	1
2	N	N
3	-	-
4	N	N
5	N	N
6	-	-
7	1	1

So f(n) = 1 + N + N + N + 1 = 2 + 3N = 3N+2
so f(n) = *O* (N) or *O* (n)

1.

```
void printFirstElementOfArray(int arr[])
{
    cout<<“First element of array :”<<arr[0];
}
```

2.

```
void printAllElementOfArray(int arr[], int size)
{
for (int i = 0; i < size; i++)
 {
     cout<< arr[i];
 }
}
```

3.

```
void printAllPossibleOrderedPairs(int arr[], int size)
{
 for (int i = 0; i < size; i++)
   {
     for (int j = 0; j < size; j++)
       {
           cout<< “(”<<arr[i]<<”,”<< arr[j]<<”)”;
       }
 }
}
```

4.

```
void printAllNumbersThenAllPairSums(int arr[], int size)
{
   for (int i = 0; i < size; i++)
   {
      cout<<, arr[i];
   }

   for (int i = 0; i < size; i++)
   {
      for (int j = 0; j < size; j++)
      {
         cout<< arr[i] + arr[j];
      }
   }
}
```

5.

```
void phytagorean(int value)
{
  for(int i = 1; i <= value; i++)
  {
          for(int j = 1; j <= value; j++)
          {
                  for(int k = 1; k <= value; k++)
                  {
```

```
                        int num1 = (i*i) + (j*j);
                        int num2 = (k*k);
                        if (num1 == num2)
                                cout<<"Pair is: (" << i << ", " << j << ", " << k << ")" << endl;
                        }
                }
        }
}
```

6.

```
void RangeCheck(int arr[],int sixe, int num1, int num2){
    int counter = 0;
    for (int i=0; i<sixe; i++)
    {
        if (arr[i] >= num1 && arr[i] <= num2)
        {
            counter++;
            cout << arr[i] << "is in the range " << endl;
        }
    }
    cout << "Total Numbers in range are: " << counter << endl;
}
```