University of Central Punjab

Faculty of Information Technology

Data Structures and Algorithms

Lab 05	
Topic	• Abstract Classes • Templates • Queues
Objective	The basic purpose of this lab is to implement ADT of queue and test its applications.

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 same as the task name. i.e. the first program should be Task_1.cpp
- **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**
- **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 void addElement(Type) = 0;

- Should add the element at the last position of the **List**

virtual Type removeElement() = 0;

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

- Write non-parameterized constructor for the above class.
- Write Copy constructor for the above class.
- Write Destructor for the above class.

Task 2

Queue: Queue is a data structure that works on First in First out (FIFO) approach. In general, items are added to the end of Array (In the way that we join at the end of a queue for a bus) and items are removed from the front of an Array. (The people at the front of the queue for the bus can get on the bus first. So using the class made in task 1, make a class named as **Queue**, having following additional functionalities:

bool empty() : Returns whether the **Queue** is empty or not. Time Complexity should be: O(1)

bool full() : Returns whether the **Queue** is full or not. Time Complexity should be: O(1)
int size() : Returns the current size of the **Queue**. Time Complexity should be: O(1)
Type front () : Returns the front element of the **Queue.** Time Complexity should be: O(1)

void enqueue(Type) : Adds the element of type Type at the top of the **Queue**. Time Complexity should be: O(1) **Hint:** You can use addElement function in it.

Type dequeue() : Deletes the first most element of the **Queue** and returns it. Time Complexity should be: O(1) **Hint:** You can use removeElement function in it. You should shift left all the elements after you remove an element from queue.

- Write non-parameterized constructor for the above class.
- Write Copy constructor for the above class.
- Write Destructor for the above class.
- Write a display function for the above class to display its all elements.

Task 3

Instantiate several objects of **Queue**, test all the functions of **Queue** on them and then display them through **display()** function.

Task 4

Now write a global function reverseElements, which should return an array that contains the reversed elements of the array received.

Hint: Use queue in this function. You can use as many queue as you like but you are not allowed use any other data structure.

Type* reverseElements (Type *arr, int size);

Create several types of arrays, test the function reverseElements on them and check whether those arrays are palindrome or not.

Task 5

Your task is to design a Queue using stacks only. You can use as many stacks as you like but you are not allowed use any other data structure.