**DSA (Data Structures & Algorithms)**

**MID-TERM EXAM**

**X2**

**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 have to work in multiple files. i.e separate .h and .cpp files**
* **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)

**Task 1:**

**Postfix Expression Evaluation:**

**Description:**

Create a program to evaluate postfix expressions. Given a valid postfix expression PostfixEvaluation(string exp), the program should calculate and display the result. If the given expression is not valid, then program should display an error message. Implement the evaluation algorithm using a stack data structure.

**Requirements:**

* Implement a stack data structure to assist in the evaluation process.
* Define functions to evaluate postfix expressions.
* Handle operators (+, -, \*, /) and operands properly during the evaluation.
* Check the expression for evaluation, if the expression is valid or a balanced expression or not.
* Display the result of the postfix expression evaluation.

**Example:**

**Input: exp = “ AB+CD\* ” OR exp = “ +AB+CD\* ” OR exp = “ AB+D+\* ”**

**Output: Invalid Expression.**

**Input: exp = “AB+CD+\*” (Consider values of A=2, B=3, C=4, D=5)**

**Output: 45**

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

**Attributes:**

1. Type \* stackArray;
2. int maxSize;
3. int stackTop;

**Functions:**

virtual void Push(Type) = 0;

virtual Type Pop() = 0;

* Using the base class (Stack), make another derived (MyStack). Implement both pure virtual functions Push () and pop() declared in base class(Stack), into derived class (MyStack).
* Implement the ‘**PostfixEvaluation’** function which is given an expression, (string exp) that validates the given postfix expression, then evaluates it and displays the result.

**Type PostfixEvaluation (string &exp)**

**OR**

**void PostfixEvaluation (string &exp)**

**Task 2:**

**Description:**

You are required to write a function called reverseLinkedList(LinkedList list). The function should reverse the elements of the LinkedList using a queue object.

The function should modify the given LinkedList in place, reversing the order of its elements.

**Example:**

**Input:**

Suppose we have data in linkedlist as follows = {1, 2, 3, 4, 5}

**Output:**

LinkedList should be modified as follows:

LinkedList = {5, 4, 3, 2, 1}

**Requirements:**

* **Function Implementation:** Write the definition of the function reverseLinkedList that reverses the elements of the given linkedlist using a queue object.
* **Queue and LinkedList Operations:** You may use standard implementations of queue and LinkedList data structures. Ensure that the necessary queue and LinkedList operations (enqueue, dequeue, insert, delete) are properly utilized.

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

**Attributes:**

1. Node \*head
2. Node \*tail

**Functions:**

virtual void insertAtTail(int) = 0;

virtual void deleteAtTail() = 0;

* Using the base class (LinkedList), make another derived (MyLinkedList). Implement both pure virtual functions **insertAtTail**(int) and **deleteAtTail**() declared in base class(LinkedList), into derived class (MyLinkedList).
* Implement the ‘**reverseLinkedList’** to modify LinkedList according to the requirement. (you can use object of LinkedList as parameter or without parameter, it’s your own choice)

**Type reverseQueue (Type &inputQueue, Type &inputLinkedList)**

**or**

**void reverseQueue ()**