**Question 1**: Two applications of Stack and Queue :

**Two Applications of Stack:**

1. Parenthesis Matching :

=> An expression will be given, we have to find out the expression is correctly matched or not.

For example: we have given an example like -

(()())() . In this expression the opening and closing parenthesis are given properly. So this is a balanced parenthesis.

If the expression is like ((()(()()) . The opening and closing parenthesis are not given properly. So this is not a balanced parenthesis.

Sample input: (()())()

Sample output: balanced

Sample input: ((()(()())

Sample output: not balanced

2. Expression Evaluation:

=> We will be given an expression. And we have to evaluate the expression and find the output of that expression.

Sample input: (5+7)\*2/8

Sample output: 3

**Two Applications of Queues:**

1. Generate Binary Numbers

=> Given a positive number n. we have to generate binary numbers between 1 to n.

Sample input: 10

Sample output: 1 10 11 100 101 110 111 1000 1001 1010 1011 1100 1101 1110 1111 10000

2. All types of customer services like bank line or ticket line are designed using the concept of queue. Because it follows the first in first out orders.

Sample input: 1 5 3 2 1 2

Sample output: 5 3 2 1 2

**Question 2:**

**CheckParenthesis.h (header file)**

//

// Created by f12r on ১২/৮/২১.

//

#ifndef ASSIGNMENT\_1\_CHECKPARENTHESIS\_H

#define ASSIGNMENT\_1\_CHECKPARENTHESIS\_H

class CheckParenthesis

{

private:

int size;

int \*list;

int top;

public:

CheckParenthesis();

CheckParenthesis(int);

void push(int);

void pop();

int topItem();

bool isEmpty();

bool isFull();

};

#endif //ASSIGNMENT\_1\_CHECKPARENTHESIS\_H

**CheckParenthesis.cpp (definition file)**

//

// Created by f12r on ১২/৮/২১.

//

#include "CheckParenthesis.h"

#include <iostream>

#include <bits/stdc++.h>

using namespace std;

CheckParenthesis::CheckParenthesis(int size){

top = -1;

this->size = size;

list = new int[size];

}

CheckParenthesis::CheckParenthesis() {

top = -1;

size = 0;

list = new int[size];

}

bool CheckParenthesis::isEmpty() {

return top==-1;

}

bool CheckParenthesis::isFull() {

return top==size-1;

}

void CheckParenthesis::push(int item) {

if (isFull()){

cout<<"list is full"<<endl;

return;

}

size++;

top++;

list[top] = item;

}

void CheckParenthesis::pop() {

if (isEmpty()){

cout<<"list is empty"<<endl;

return ;

}

top--;

size--;

}

int CheckParenthesis::topItem() {

return list[top];

}

**main.cpp (driver file)**

#include "CheckParenthesis.h"

#include "CheckParenthesis.cpp"

#include <iostream>

using namespace std;

int main()

{

string expression;

cout << "Input: ";

// input expression

cin >> expression;

int len = 0;

// count how many parenthesis in the mathemetical expression

for (int i = 0; i < expression.size(); i++)

{

if (expression[i] == '(' || expression[i] == ')')

{

len++;

}

}

CheckParenthesis checkParenthesis(len);

cout << "Output: ";

for (int i = 0; i < expression.size(); i++)

{

if (expression[i] == '(')

{

checkParenthesis.push(expression[i]);

}

else if (expression[i] == ')')

{

if (checkParenthesis.isEmpty())

{

cout << "Not Balanced" << endl;

return 0;

}

checkParenthesis.pop();

}

else

{

continue;

}

}

if (checkParenthesis.isEmpty())

{

cout << "Balanced" << endl;

}

else

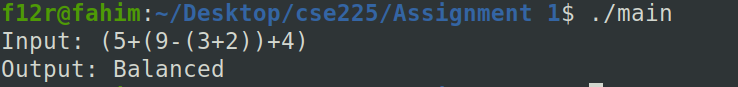
{

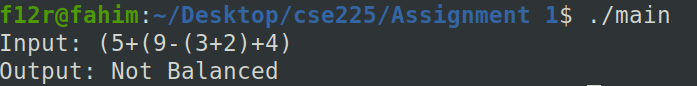
cout << "Not Balanced" << endl;

}

}

**Output :**

****

****