**K23-0607 DS LAB # 6 Sept,30 2024**

**Question # 1**

#include<iostream>

using namespace std;

class stack{

    public :

    int top;

    int size;

    char \*arr;

    stack(int s){

        size = s;

        top = -1;

        arr = new char [size];

    }

    stack(){

        size = -1;

        top = -1;

        arr = nullptr;

    }

    void push(char val){

        if( size - top > 1){

            top++;

            arr[top] = val;

        }

        else {

            cout << "Stack Overflow " << endl;

        }

    }

    void pop(){

        if( top != -1 ){

           top--;

        }

        else{

            cout << "Stack Underflow " << endl;

        }

    }

    char peek(){

        if( top !=- 1 ){

            return arr[top];

        }

        else{

            cout << "Stack is Empty " << endl;

            return -1;

        }

    }

    bool empty(){

        if ( top == -1) return false;

        return true;

    }

    ~stack(){

        delete []arr;

    }

};

bool isPalindrome(stack &s){

    char \*temp = new char [s.size];

    int index =  0;

    while( !s.empty() ){

        temp[index++] = s.peek();

        s.pop();

    }

    int start = 0;

    int end = s.size;

    while( start < end){

        if( temp[start] != temp[end] ){

            delete []temp;

            return false;

        }

        start++;

        end--;

    }

    delete []temp;

    return true;

}

int main(){

    string word = "BORROWROB";

    stack s(word.size());

    s.push('B');

    s.push('O');

    s.push('R');

    s.push('R');

    s.push('O');

    s.push('W');

    s.push('R');

    s.push('O');

    s.push('B');

    bool found = isPalindrome(s);

    if( found ) {

        cout << "It is a Palindrom : " << word << endl;

    }

    else{

        cout << "Not a Palindrome : " << word << endl;

    }

}

A blue and black screen

Description automatically generated with medium confidence

**Question # 2**

#include <iostream>

using namespace std;

class Queue {

public:

int front;

int rear;

int \*arr;

int size;

Queue(int s) {

front = 0;

rear = 0;

size = s;

arr = new int[size];

}

void enqueue(int val) {

if (rear < size) {

arr[rear++] = val;

} else {

cout << "Queue Overflow" << endl;

}

}

void dequeue() {

if (front < rear) {

front++;

} else {

cout << "Queue is Empty" << endl;

}

}

bool empty() {

return front == rear;

}

int Front() {

if (front < rear) {

return arr[front];

} else {

cout << "Queue is Empty" << endl;

return -1;

}

}

};

int main() {

int arr[7] = {13, 7, 4, 1, 6, 8, 10};

Queue q(7);

for (int i = 0; i < 7; i++) {

q.enqueue(arr[i]);

}

int i = 1;

while (!q.empty()) {

cout << "CheckOut " << i << " : " << q.Front() << endl;

q.dequeue();

i++;

}

delete[] q.arr;

}

A screen shot of a computer

Description automatically generated

**Question # 3**

#include <iostream>

#include <string>

#include <cctype>

#include <cmath>

using namespace std;

struct Node {

    float val;

    Node\* next;

};

class Stack {

private:

    Node\* top;

public:

    Stack() {

        top = nullptr;

    }

    void push(float value) {

        Node\* newNode = new Node();

        newNode->val = value;

        newNode->next = top;

        top = newNode;

    }

    float pop() {

        if (top == nullptr) {

            return -1;

        }

        Node\* temp = top;

        top = top->next;

        float popp = temp->val;

        delete temp;

        return popp;

    }

    float peek() {

        if (top == nullptr) {

            return -1;

        }

        return top->val;

    }

    bool isEmpty() {

        return top == nullptr;

    }

};

float perform(float x, float y, char ch) {

    switch (ch) {

        case '\*': return x \* y;

        case '/': return x / y;

        case '^': return pow(x, y);

        case '+': return x + y;

        case '-': return x - y;

        default: return 0;

    }

}

bool isOperator908(char ch) {

    return (ch == '+' || ch == '-' || ch == '\*' || ch == '/' || ch == '^');

}

int prece(char c) {

    if (c == '^') return 3;

    else if (c == '\*' || c == '/') return 2;

    else if (c == '+' || c == '-') return 1;

    else return -1;

}

string Logic(string check) {

    Stack st;

    string postfix;

    int i = 0;

    while (i < check.length()) {

        char c = check[i];

        if (isdigit(c) || c == '.') {

            while (i < check.length() && (isdigit(check[i]) || check[i] == '.')) {

                postfix += check[i];

                i++;

            }

            postfix += ' ';

            continue;

        }

        if (c == '(') {

            if (i > 0 && (isdigit(check[i - 1]) || check[i - 1] == ')')) {

                postfix += '\*';

                postfix += ' ';

            }

            st.push(c);

        }

        else if (c == ')') {

            while (!st.isEmpty() && st.peek() != '(') {

                postfix += st.pop();

                postfix += ' ';

            }

            st.pop();

        }

        else if (isOperator908(c)) {

            while (!st.isEmpty() && prece(c) <= prece(st.peek())) {

                postfix += st.pop();

                postfix += ' ';

            }

            st.push(c);

        }

        i++;

    }

    while (!st.isEmpty()) {

        postfix += st.pop();

        postfix += ' ';

    }

    return postfix;

}

float solve(string postfix) {

    Stack st;

    int i = 0;

    while (i < postfix.length()) {

        char c = postfix[i];

        if (isdigit(c) || c == '.') {

            string num;

            while (i < postfix.length() && (isdigit(postfix[i]) || postfix[i] == '.')) {

                num += postfix[i];

                i++;

            }

            st.push(stof(num));  // string  To float

        }

        else if (isOperator908(c)) {

            float b = st.pop();

            float a = st.pop();

            float result = perform(a, b, c);

            st.push(result);

        }

        i++;

    }

    return st.pop();

}

int main() {

    string infix = "12+13-5\*(0.5+0.5)+1";

    string postfix = Logic(infix);

    cout << "Postfix expression: " << postfix << endl;

    float res = solve(postfix);

    cout << "Result: " << res << endl;

}

A computer screen with text

Description automatically generated

**END**