**Answer Script**

Basic Data Structures and Problem Solving Part-II

**Week-7, Module 27: Lab Assignment 2**

(nayeem.cse6.bu@gmail.com)

| **Question No. 01** |
| --- |
| Implement a template based Queue using a dynamic array which supports the enqueue, dequeue and front operations. |
| Answer No. 01 |

**Implementing below a template based Queue using dynamic array which supports the enqueue, dequeue and front operations:**

```
#include <bits/stdc++.h>

using namespace std;

template<class T>
class Queue {
public:
    T *a;
    T array_cap;
    T l, r;
    int sz;

    Queue() {
        a = new T[1];
        array_cap = 1;
        l = 0;
        r = -1;
        sz = 0;
    }

    void remove_circular() {
        if(l > r) {
```

```
            T *tmp = new T[array_cap];
            T idx = 0;

            for(int i = l; i <= array_cap; i++) {
                tmp[idx] = a[i];
                idx++;
            }

            for(int i = 0; i < r; i++) {
                tmp[idx] = a[i];
                idx++;
            }
            swap(tmp, a);
            l = 0;
            r = array_cap - 1;
        }
    }

    // increase size
    void increase_size() {
        remove_circular();
        T *tmp = new T[array_cap*2];
        for(int i = 0; i < array_cap; i++) {
            tmp[i] = a[i];
        }
        swap(a, tmp);
        array_cap = array_cap * 2;
        delete []tmp;
    }

    // Insert Element O(1)
    void enqueue(T val) {
        if(sz == array_cap) {
            increase_size();
        }
        r++;
        if(r == array_cap) {
            r = 0;
```

```
        }

        a[r] = val;
        sz++;
    }

    // Delete Element O(1)
    void dequeue() {
        if(sz == 0) {
            cout << "Queue is empty!\n";
            return;
        }
        l++;
        if(l == array_cap) {
            l = 0;
        }
        sz--;
    }

    // Return the front Element O(1)
    T front() {
        if(sz == 0) {
            cout << "Queue is empty!\n";
            return -1;
        }
        return a[l];
    }

    // Return the queue size O(1)
    int size() {
        return sz;
    }
};

// driver code
int main() {

    Queue<int> q;
```

```
    q.enqueue(5);
    q.enqueue(6);
    q.enqueue(7);

    cout << "Queue Size = " << q.size() << "\n";
    cout << "Front Element = " << q.front() << "\n";

    cout << "\n";

    q.enqueue(8);
    cout << "Queue Size [After Enqueued] = " << q.size() << "\n";
    cout << "Front Element = " << q.front() << "\n";

    q.dequeue();
    cout << "\n";

    cout << "Queue Size [After Dequeued] = " << q.size() << "\n";
    cout << "Front Element = " << q.front() << "\n";

return 0;
}
```

| **Question No. 02** |
| --- |
| Implement Template based Stack using a singly linked-list. |
| Answer No. 02 |

**Implementing below the template based stack using singly linked list:**

```
#include <bits/stdc++.h>

using namespace std;
```

```
template <class T>
class node {
public:
    T data;
    node * nxt;
};

template <class T>
class SinglyLinkedList {
public:
    node<T> * head;
    int sz;
    SinglyLinkedList() {
        head = NULL;
        sz = 0;
    }

    // function to create a new node with given data and insert it
O(1)
    node<T> * CreateNewNode(T data) {
        node<T> *newnode = new node<T>;
        newnode->data = data;
        newnode->nxt = NULL;
        return newnode;
    }

    // Insert at head with given data O(1)
    void InsertAtHead(T data) {
        sz++;
        node<T> *newnode = CreateNewNode(data);
        if(head == NULL) {
            head = newnode;
            return;
        }
        node<T> *a = head;
        newnode->nxt = a;
        head = newnode;
    }
```

```
    // Delete at head
    void DeleteAtHead() {
        if(head == NULL) {
            return;
        }
        sz--;
        node<T> *a = head;
        head = a->nxt;
        delete a;
    }


    // print the size of the linked list O(1)
    int getSize() {
        return sz;
    }
};

// Stack using singly linked list
template <class T>
class Stack {
public:
    SinglyLinkedList<T> sl;

    Stack() {

    }

    T top() {
        if(sl.getSize() == 0) {
            cout << "Stack is empty!\n";
            T a;
            return a;
        }
        return sl.head->data;
    }
```

```
    void push(T val) {
        sl.InsertAtHead(val);
    }

    void pop() {
        if(sl.getSize() == 0) {
            cout << "Stack is empty!\n";
            return;
        }
        sl.DeleteAtHead();
    }
};


int main() {

    Stack<double> sl;
    sl.push(4.5);
    sl.push(6.7);
    sl.push(8.7);

    cout << sl.top() << "\n";

    sl.pop();

    cout << sl.top() << "\n";

    Stack<char> sl2;
    sl2.push('a');
    sl2.push('b');
    sl2.push('c');

    cout << sl2.top() << "\n";

    sl2.pop();

    cout << sl2.top() << "\n";
```

```
return 0;
}
```

## Question No. 03

Write a program to convert an infix expression to a postfix expression. The expression will contain the following characters [ a-z , + , - , * , / , ( , ) ].

| Sample Input | Sample Output |
|---|---|
| a+(b+c)*d-e | abc+d*+e- |
| (a+b)*(c+d) | ab+cd+* |

## Answer No. 03

**C++ program to convert an infix expression to a postfix expression:**

```
#include <bits/stdc++.h>

using namespace std;

int precedence(char ch) {
    if(ch == '+' || ch == '-') {
        return 0;
    }
    else if(ch == '*' || ch == '/') {
        return 1;
    }
    return -1;
}

int main() {
    string str;
    cin >> str;
```

```
    string ans = "";
    stack<char> stk;
    for(int i = 0; i < str.size(); i++) {
        char ch = str[i];

        if(ch >= 'a' && ch <= 'z') {
            ans += ch;
        }

        else if(ch == '(') {
            stk.push(ch);
        }

        else if(ch == ')') {
            while(stk.size() && stk.top() != '(') {
                ans += stk.top();
                stk.pop();
            }
            stk.pop();
        }

        else {
                        while(stk.size()  &&  precedence(stk.top())  >=
precedence(ch)) {
                ans += stk.top();
                stk.pop();
            }

            stk.push(ch);
        }
    }

    while(stk.size()) {
        ans += stk.top();
        stk.pop();
    }

    cout << ans << "\n";
```

```
    return 0;
}
```

## Question No. 04

Evaluate it using stack. All the numbers are single digit numbers in the input so you don't have to worry about multi digit numbers.

| Sample Input | Sample Output |
|---|---|
| 4+(5+6)*8-1 | 91 |
| (2+4)*(5+6) | 66 |

Congratulations you just built a mini calculator if you solved it correctly.

## Answer No. 04

**Implementing the above problem using stack:**

```
#include <bits/stdc++.h>
using namespace std;

int eval(int op1, int op2, char op)
{
    switch(op)
    {
        case '+': return op1 + op2;
        case '-': return op1 - op2;
        case '*': return op1 * op2;
        case '/': return op1 / op2;
        default: return 0;
    }
}

int postfixEval(string exp)
{
    stack<int> s;
```

```
    for (int i = 0; i < exp.length(); i++)
    {
        if (isdigit(exp[i]))
        {
            int operand = 0;
            while (i < exp.length() && isdigit(exp[i]))
            {
                operand = (operand * 10) + (exp[i] - '0');
                i++;
            }
            i--;
            s.push(operand);
        }
        else
        {
            int op2 = s.top();
            s.pop();
            int op1 = s.top();
            s.pop();

            int result = eval(op1, op2, exp[i]);
            s.push(result);
        }
    }
    return s.top();
}

int main()
{
    string exp;
    cin >> exp;
    cout << postfixEval(exp) << endl;
    return 0;
}
```

| Question No. 05 |
| --- |
| Implement Template based Deque using a doubly linked-list which supports push_front, push_back, pop_back, pop_front, front, back operations. |
| Answer No. 05 |

**Implementing below template based deque using a doubly linked list which supports push_front, push_back, pop_back, pop_front, front, back operations:**

```
#include <bits/stdc++.h>

using namespace std;

template<class T>
class node {
public:
    T data;
    node* prv;
    node* nxt;
};

template<class T>
class Deque {
public:
    node<T>* head;
    node<T>* tail;
    int sz;

    Deque() {
        head = NULL;
        tail = NULL;
        sz = 0;
    }

    node<T> * CreateNewNode(T value) {
        node<T> * newnode = new node<T>;
```

```
        newnode->data = value;
        newnode->nxt = NULL;
        newnode->prv = NULL;
        return newnode;
    }

    // push_back O(1)
    void push_back(T val) {
        node<T>* newnode = CreateNewNode(val);
        if(sz == 0) {
            head = newnode;
            tail = newnode;
            sz++;
            return;
        }
        tail->nxt = newnode;
        newnode->prv = tail;
        tail = newnode;
        sz++;
    }

    // push front O(1)
    void push_front(T val) {
        node<T> * newnode = CreateNewNode(val);
        if(sz == 0) {
            head = newnode;
            tail = newnode;
            sz++;
            return;
        }
        head->prv = newnode;
        newnode->nxt = head;
        head = newnode;
        sz++;
        return;
    }

    // pop back O(1)
```

```
    void pop_back() {
        if(sz == 0) {
            cout << "Deque is empty!\n";
            return;
        }

        if(sz == 1) {
            delete tail; // or delete head;
            head = NULL;
            tail = NULL;
            sz--;
            return;
        }
        node<T>* a = tail;
        tail = tail->prv;
        delete a;
        tail->nxt = NULL;
        sz--;
    }

    // pop front
    void pop_front() {
        if(sz == 0) {
            cout << "Deque is empty!\n";
            return;
        }
        if(sz == 1) {
            delete head; // or delete tail
            head = NULL;
            tail = NULL;
            sz--;
            return;
        }
        node<T>* a = head;
        head = head->nxt;
        delete a;
        head->prv = NULL;
        sz--;
```

```
    }

    // front element O(1)
    int front() {
        if(sz == 0) {
            cout << "Deque is empty!\n";
            return -1;
        }
        return head->data;
    }

    // back element O(1)
    int back() {
        if(sz == 0) {
            cout << "Deque is empty\n";
            return -1;
        }
        return tail->data;
    }
};

// driver code
int main() {

    Deque<int> d;
    d.push_back(5); //5
    d.push_front(7); // 7 5
    cout << d.front() << "\n"; // output: 7
    d.pop_back(); // 7
    d.push_back(3); // 7 3
    cout << d.back() << "\n"; // output: 3
    d.pop_front(); // 3
    cout << d.back() << "\n"; // output: 3

return 0;
}
```

## Question No. 06

Given a string, check if it's a palindrome using a Deque.

| Sample Input | Sample Output |
|---|---|
| abcba | Yes |
| abcca | No |

Hint: Check the first and last character. If they are equal then pop them and continue this process until the string becomes empty.

## Answer No. 06

**C++ Program to check a number palindrome or not using deque:**

```
#include <bits/stdc++.h>

using namespace std;

int main() {

    string str;
    cin >> str;

    deque<char> dq;
    for(int i = 0; i < str.size(); i++) {
        dq.push_back(str[i]);
    }

    while(dq.size() > 1) {

       if(dq.front() != dq.back()) {
            cout << "No\n";
            return 0;
       }

       dq.pop_front();
```

```
        dq.pop_back();
    }

    cout << "Yes\n";

return 0;
}
```

| Question No. 07 |
| --- |
| Write a function **void deleteValue(list<int> & l , int value)'** -> This function will delete the first occurrence of the element that is equal to the input **value** from the stl list.<br><br>**Sample Input:** STL list containing [7, 3, 8, 4, 5, 4], value : 4<br>**Sample Output:** STL list containing [7, 3, 8, 5, 4] |
| Answer No. 07 |

**Implementing below the function void deleteValue(list<int> &l, int value) to delete the first occurrence of the element that is equal to the input value from the stl list:**

```
#include <bits/stdc++.h>

using namespace std;

// function to delete the first occurrence of the value
void deleteValue(list<int> &l , int value) {
    for(auto it = l.begin(); it != l.end(); it++) {
        if(*it == value) {
            l.erase(it);
            break;
        }
    }
```

```
}

// function to print the linked list
void print(list<int> &l){
    for(auto it = l.begin(); it != l.end(); it++) {
        cout << *it << " ";
    }
}

// driver code
int main() {

    list<int> l = {7, 3, 8, 4, 5, 4};
    deleteValue(l, 4);
    print(l);

    return 0;
}
```