Answer Script

Question No. 01

Implement a template based Queue using a dynamic array which supports the enqueue, dequeue and front operations.

Answer No. 01

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

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

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

   void Resize()
   {
     cap = cap * 2;
     T *new_a = new T[cap];
     for(int i=l; i<=r; i++)
        new_a[i] = a[i];
     swap(a, new_a);
     delete [] new_a;
   }

  void Enqueue (T value)
  {
```

```
        if(r+1 >= cap){Resize();}
        r++;
        a[r] = value;
        sz++;
    }

    void Dequeue()
    {
        if(l > r){cout<<"ALREADY EMPTY!!"<<endl;return;}
        else {l++; sz--;}
    }

    void front()
    {
        if(l > r){cout<<"EMPTY!!"<<endl;return;}
        cout<<a[l]<<endl;
    }

    int size(){return sz;}

};

int main()
{
    Queue<int> que;
    que.Enqueue(10);
    que.Enqueue(20);
    que.Enqueue(30);
    que.Enqueue(40);
    que.Enqueue(50);
    que.Enqueue(60);
    que.Enqueue(70);
    que.Enqueue(80);
    que.Enqueue(90);

    cout<<"SIZE AFTER ENQUEUE : "<<que.size()<<endl;
    int sz = que.size();
    for(int i=0; i<sz; i++){
        que.front();
        que.Dequeue();
    }
```

```
    cout<<"SIZE AFTER DEQUEUE : "<<que.size()<<endl;

    Queue<char> que2;
    que2.Enqueue('A');
    que2.Enqueue('B');
    que2.Enqueue('C');
    que2.Enqueue('D');
    que2.Enqueue('E');
    que2.Enqueue('F');

    cout<<endl;
    cout<<endl;

    cout<<"SIZE AFTER ENQUEUE : "<<que2.size()<<endl;
    int sz2 = que2.size();
    for(int i=0; i<sz2; i++){
        que2.front();
        que2.Dequeue();
    }
    cout<<"SIZE AFTER DEQUEUE : "<<que2.size()<<endl;

    return 0;
}
```

Question No. 02

Implement Template based Stack using a singly linked-list.

Answer No. 02

```
#include<bits/stdc++.h>
using namespace std;
template<class T>
class node{
public:
    T data;
    node<T> *next;
};
template<class T>
class SLL{
public:
```

```
    node<T> *head;
    int sz;

    SLL()
    {
        head = NULL;
        sz = 0;
    }

    node<T> *CreateNode(T value)
    {
        sz++;
        node<T> *newnode = new node<T>;
        newnode->data = value;
        newnode->next = NULL;
        return newnode;
    }
    void InsertAtHead(T value)
    {
        node<T> *temp = CreateNode(value);
        if(head == NULL){head = temp;}
        else
        {
            temp->next = head;
            head = temp;
        }
    }
    void DeleteAtHead()
    {
        node<T> *temp = head;
        temp = head->next;
        swap(head, temp);
        delete temp;
        sz--;
    }
    int Size()
    {
        return sz;
    }
};
template<class T>
```

```
class Stack{
public:
   SLL<T> sl;

   Stack()
   {

   }

   void Push(T value)
   {
      sl.InsertAtHead(value);
   }

   void Pop()
   {
      if(sl.Size()<=0){cout<<"Already Empty"<<endl;return;}
      else{sl.DeleteAtHead();}
   }
   void Top()
   {
      if(sl.Size()==0){cout<<"Stack is Empty"<<endl;return;}
      else{cout<<sl.head->data<<endl;}
   }
};

int main()
{
   Stack<int> st;
   st.Top();
   st.Push(10);
   st.Top();
   st.Push(20);
   st.Top();
   st.Push(30);
   st.Top();

   st.Pop();
   st.Top();

   st.Pop();
```

```
    st.Pop();
    st.Pop();
    st.Top();

    cout<<endl;

    Stack<char> st2;
    st2.Top();
    st2.Push('A');
    st2.Top();
    st2.Push('B');
    st2.Top();
    st2.Push('C');
    st2.Top();

    st2.Pop();
    st2.Top();

    st2.Pop();
    st2.Pop();
    st2.Pop();
    st2.Top();

    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

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

```
using namespace std;

int precedence(char op)
{
   if (op == '+' || op == '-')
      return 1;
   if (op == '*' || op == '/')
      return 2;
   if (op == '^')
      return 3;
   return 0;
}

int main()
{
   string expression;
   cin>>expression;
   stack<char> s;
   string postfix = "";

   for (int i = 0; i < expression.length(); i++)
   {
      char c = expression[i];

      if (c == ' ')
         continue;

      if (isdigit(c) || isalpha(c))
      {
         postfix += c;
      }
      else if (c == '(')
      {
         s.push(c);
      }
      else if (c == ')')
      {
         while (!s.empty() && s.top() != '(')
         {
            postfix += s.top();
            s.pop();
```

```
        }
        if (!s.empty() && s.top() == '(')
            s.pop();
    }
    else
    {
        while (!s.empty() && precedence(c) <= precedence(s.top()))
        {
            postfix += s.top();
            s.pop();
        }
        s.push(c);
    }
  }

  while (!s.empty())
  {
    postfix += s.top();
    s.pop();
  }

  cout<<postfix<<endl;
  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

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

```
int precedence(char op)
{
   if (op == '+' || op == '-')
      return 1;
   if (op == '*' || op == '/')
      return 2;
   if (op == '^')
      return 3;
   return 0;
}

int evaluatePostfix(string expression)
{
   stack<int> s;

   for (int i = 0; i < expression.length(); i++)
   {
      char c = expression[i];

      if (isdigit(c))
      {
         s.push(c - '0');
      }
      else
      {
         int operand2 = s.top();
         s.pop();
         int operand1 = s.top();
         s.pop();

         int result;
         switch (c)
         {
         case '+':
            result = operand1 + operand2;
            break;
         case '-':
            result = operand1 - operand2;
            break;
         case '*':
```

```
            result = operand1 * operand2;
            break;
         case '/':
            result = operand1 / operand2;
            break;
         }

         s.push(result);
      }
   }

   return s.top();
}

string infixToPostfix(string expression)
{
   stack<char> s;
   string postfix = "";

   for (int i = 0; i < expression.length(); i++)
   {
      char c = expression[i];

      if (c == ' ')
         continue;

      if (isdigit(c))
      {
         postfix += c;
      }
      else if (c == '(')
      {
         s.push(c);
      }
      else if (c == ')')
      {
         while (!s.empty() && s.top() != '(')
         {
            postfix += s.top();
            s.pop();
         }
```

```
        if (!s.empty() && s.top() == '(')
            s.pop();
      }
      else
      {
         while (!s.empty() && precedence(c) <= precedence(s.top()))
         {
            postfix += s.top();
            s.pop();
         }
         s.push(c);
      }
   }

   while (!s.empty())
   {
      postfix += s.top();
      s.pop();
   }

   return postfix;
}

int main()
{
   string expression;
   cin>>expression;
   string postfix = infixToPostfix(expression);
   cout <<evaluatePostfix(postfix)<< 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

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

```
template<class T>
class node{
public:
   node<T>* prev;
   T data;
   node<T>* next;
};
template<class T>
class DLL{
public:
   node<T> *head;
   node<T> *tail;
   int sz;

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

   node<T> *CreateNode(T value)
   {
      sz++;
      node<T> *newnode = new node<T>;
      newnode->prev = NULL;
      newnode->data = value;
      newnode->next = NULL;
      return newnode;
   }

   void InsertAtHead(T value)
   {
      node<T>* temp = CreateNode(value);
      if(head==NULL)
      {
         head = temp;
         tail = temp;
      }
      else
      {
```

```
        temp->next = head;
        head->prev = temp;
        head = temp;
    }
}
void InsertAtTail(T value)
{
    node<T> *temp = CreateNode(value);
    if(head==NULL)
    {
        head = temp;
        tail = temp;
    }
    else
    {
        tail->next = temp;
        temp->prev = tail;
        tail = temp;
    }
}
void DeleteAtHead()
{
    node<T> *temp = head;
    if(head == NULL){return;}
    else
    {
        temp = head->next;
        swap(head, temp);
        delete temp;
        sz--;
    }
}
void DeleteAtTail()
{
    if(tail == NULL){return;}
    else
    {
        node<T> *temp = tail;
        temp = tail->prev;
        tail = temp;
        sz--;
```

```
        }
    }
    T front()
    {
        if(sz==0){return 0;}
        return head->data;
    }
    T back()
    {
        if(sz==0){return 0;}
        return tail->data;
    }

};
template<class T>
class Deque{
public:
    DLL<T> dl;
    int sz;
    Deque()
    {
        sz = 0;
    }
    void Push_front(T value)
    {
        sz++;
        dl.InsertAtHead(value);
    }
    void Pop_front()
    {
        if(sz==0){cout<<"Already Empty"<<endl;return;}
        else
        {
            dl.DeleteAtHead();
            sz--;
        }
    }
    void Push_back(T value)
    {
        sz++;
        dl.InsertAtTail(value);
```

```
    }
    void Pop_back()
    {
        if(sz==0){cout<<"Already Empty"<<endl;return;}
        else
        {
            dl.DeleteAtTail();
            sz--;
        }
    }
    void Front()
    {
        if(sz==0){cout<<"Empty"<<endl;return;}
        cout<<dl.front()<<endl;
    }
    void Back()
    {
        if(sz==0){cout<<"Empty"<<endl;return;}
        cout<<dl.back()<<endl;
    }
    int Size()
    {
        return sz;
    }
};



int main()
{
    Deque<int> dq;
    dq.Push_back(30);
    dq.Push_back(10);
    dq.Push_front(20);
    dq.Push_back(40);
    dq.Push_front(80);
    cout<<dq.Size()<<endl;
    dq.Front();
    dq.Back();
    dq.Pop_front();
    dq.Pop_front();
```

```
    dq.Pop_front();
    dq.Pop_front();
    dq.Pop_back();
    dq.Front();
    dq.Back();
    cout<<dq.Size()<<endl;

    Deque<char> dq2;
    dq2.Push_back('A');
    dq2.Push_back('B');
    dq2.Push_front('C');
    dq2.Push_back('D');
    dq2.Push_front('E');
    cout<<dq2.Size()<<endl;
    dq2.Front();
    dq2.Back();
    dq2.Pop_front();
    dq2.Pop_front();
    dq2.Pop_front();
    dq2.Pop_front();
    dq2.Pop_back();
    dq2.Front();
    dq2.Back();
    cout<<dq.Size()<<endl;
}
```

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

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

```
using namespace std;

int main()
{
   string st;cin>>st;
   deque<char> dq;

   for(int i=0; i<st.size(); i++){
      dq.push_back(st[i]);
   }
   for(int j=0; j<st.size()/2; j++)
   {
      if(dq.front() == dq.back())
      {
         dq.pop_front();
         dq.pop_back();
      }
      else
      {
         cout<<"NO"<<endl;
         return 0;
      }
   }
   cout<<"YES"<<endl;
   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.

Sample Input: STL list containing [7, 3, 8, 4, 5, 4], value : 4
Sample Output: STL list containing [7, 3, 8, 5, 4]

Answer No. 07

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

```
void deleteValue(list<int> &l, int value)
{
   for(auto it = l.begin(); it!=l.end(); it++){
      if(*it == value){
         l.erase(it);
         break;
      }
   }
   for(auto it = l.begin(); it != l.end(); it++){
      cout<<*it<<" ";
   }
}

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