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 wish>
class Queue
{
public:
   wish *A;
   int cap;
   int sz;
   int l;
   Queue()
   {
      A=new wish[1];
      cap=1;
      sz=0;
      l=0;
   }
   void increase_size()
   {
      wish *T=new wish[cap*2];
      for(int i=0;i<cap;i++)
      {
         T[i]=A[i];
      }
      swap(T,A);
      cap*=2;
      delete []T;
   }
   void enqueue(wish value)
   {
      if(cap==sz)
      {
         increase_size();
      }
```

```
        A[sz]=value;
        sz++;
    }
    void dequeue()
    {
        if(sz==0)
        {
            cout<<"Queue is empty\n";
            return;
        }
        l++;
        sz--;

    }
    int front()
    {
        return A[l];
    }
};
int main()
{
    int n;cin>>n;
    Queue<int>q;
    for(int i=1;i<=n;i++)
    {
        int a;cin>>a;
        q.enqueue(a);
    }
    cout<<q.front()<<" ";
    q.dequeue();
    cout<<q.front()<<" ";
    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 wish>
class node
{
public:
   wish data;
   node *nxt;
};
template< class wish>
class Linked_List
{
public:
   node<wish> *head;
   int sz;
   Linked_List()
   {
      head = NULL;
      sz=0;
   }
   node<wish> *Create_New_Node(wish value)
   {
      node<wish>*newnode;
      newnode = new node<wish>;
      newnode->data = value;
      newnode->nxt = NULL;
      return newnode;
   }
   void Insert_At_Head(int value)
   {
      sz++;
      node<wish> *a = Create_New_Node(value);
      if(head == NULL)
      {
         head = a;
         return;
      }
      a->nxt = head;
      head = a;
   }
   void Traverse()
   {
      node<wish>* a = head;
```

```
    while(a!= NULL)
    {
        cout<<a->data<<" ";
        a = a->nxt;
    }
    cout<<"\n";
}
int Get_index(int value)
{
    node<wish>* a = head;
    int index = 0;
    while(a!= NULL)
    {
        if(a->data==value)
        {
            return index;
        }
        a = a->nxt;
        index++;
    }
    return -1;
}
void Search_All_Value(int value)
{
    node<wish>* a = head;
    int index = 0;
    while(a!= NULL)
    {
        if(a->data==value)
        {
            cout<<value<<" is found at index "<<index<<"\n";
        }
        a = a->nxt;
        index++;
    }
}
int get_Size()
{
    return sz;
}
int get_index_value(int index)
{
    node<wish>* a = head;
```

```
    int id=0;
    while(a!=NULL)
    {
        if(id==index)
        {
            return a->data;
        }
        a=a->nxt;
        id++;
    }
}
void Insert_At_Any_Index(int index, int value)
{
    if(index <0 || index > sz)
    {
        return;
    }
    if(index==0)
    {
        Insert_At_Head(value);
        return;
    }
    sz++;
    node<wish>* a = head;
    int cur_index = 0;
    while(cur_index!=index-1)
    {
        a = a->nxt;
        cur_index++;
    }
    node<wish>*newnode = Create_New_Node(value);
    newnode->nxt = a->nxt;
    a->nxt = newnode;
}
int Last()
{
    return head->data;
}
void Delete_At_Head()
{
    if(head == NULL)
    {
        return;
```

```
    }
    sz--;
    node<wish>* a = head;
    head = a->nxt;
    delete a;
}
void Delete_Any_Index(int index)
{
    if(index <0 || index > sz-1)
    {
        return;
    }
    if(index==0)
    {
        Delete_At_Head();
        return;
    }
    sz--;
    node<wish>* a = head;
    int cur_index = 0;
    while(cur_index != index-1)
    {
        a = a->nxt;
        cur_index++;
    }
    node<wish> *b = a->nxt;
    a->nxt = b->nxt;
    delete b;
}

void Insert_After_Value(int value , int data)
{
    node<wish>* a = head;
    while(a != NULL)
    {
        if(a->data == value)
        {
            break;
        }
        a = a->nxt;
    }
    if(a== NULL)
    {
```

```
        cout<<value<<" doesn't exist in linked-list.\n";
        return;
    }
    sz++;
    node<wish> *newnode = Create_New_Node(data);
    newnode->nxt = a->nxt;
    a->nxt = newnode;
}
void print_reverse()
{
    Reverse(head);
    cout<<"\n";
}
void Delete_linked_list()
{
    while(head!=NULL)
    {
        node<wish>* a = head;
        head=head->nxt;
        delete a;
    }
}
void Delete_zero()
{
    vector<int>A;
    node<wish>* a = head;
    while(a!=NULL)
    {
        if(a->data!=0)
        {
            A.push_back(a->data);
        }
        a=a->nxt;
    }
    Delete_linked_list();
    for(int i=A.size()-1;i>=0;i--)
    {
        Insert_At_Head(A[i]);
    }
}
int Odd_index_node_sum()
{
    int sum=0;
```

```
    node<wish>* a = head;
    int id=0;
    while(a!=NULL)
    {
        if(id%2!=0)
        {
            sum+=a->data;
        }
        a=a->nxt;
        id++;
    }
    return sum;
}
bool has_duplicate()
{
    vector<wish>A;
    node<wish>* a = head;
    while(a!=NULL)
    {
        A.push_back(a->data);
        a=a->nxt;
    }
    sort(A.begin(),A.end());
    for(int i=1;i<A.size();i++)
    {
        if(A[i-1]==A[i])
        {
            return true;
        }
    }
    return false;

}
int get_Last()
{
    node<wish>* a = head;
    while(a->nxt!=NULL)
    {
        a=a->nxt;
    }
    return a->data;
}
double get_Average()
```

```
    {
        node<wish>* a = head;
        double b=0;
        while(a!=NULL)
        {
            b+=(double)a->data;
            a=a->nxt;
        }
        return b/sz;
    }
    private:
    void Reverse(node<wish>*a)
    {
        if(a==NULL)
        {
            return;
        }
        Reverse(a->nxt);
        cout<<a->data<<" ";
    }

};
template< class wish>
class Stack{
public:
    Linked_List<wish>L;
    int sz;
    Stack()
    {

    }
    void push(wish Value)
    {
        L.Insert_At_Head(Value);
    }
    void pop()
    {
        L.Delete_At_Head();
    }
    int top()
    {
        return L.Last();
    }
```

```
};
int main()
{
   Stack<int>st;
   int n;
   cin>>n;
   for(int i=1;i<=n;i++)
   {
      int a;cin>>a;
      st.push(a);
   }
   cout<<st.top()<<" ";
   st.pop();
   cout<<st.top()<<" ";
   st.pop();
   cout<<st.top()<<" ";
}
```

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 prec(char c)
{
   if(c == '+' || c == '-')
      return 1;
   else if(c == '*' || c == '/')
      return 2;
   else
```

```
    return 0;
}
int main()
{
   string R;
   cin>>R;
   stack<char>st;
   string p;
   for(int i = 0; i < R.size(); i++)
   {
      if(R[i] >= 'a' && R[i] <= 'z')
         p+=R[i];
      else if(R[i] == '(')
         st.push('(');
      else if(R[i] == ')')
      {
         while(st.size()!=0 && st.top() != '(')
         {
            p+=st.top();
            st.pop();
         }
         if(st.top() == '(')
         {
            st.pop();
         }
      }
      else{
         while(st.size() && prec(R[i]) <= prec(st.top()))
         {
            p+=st.top();
            st.pop();
         }
         st.push(R[i]);
      }
   }
   while(st.size()!=0')
   {
      p+= st.top();
      st.pop();
   }
   cout<<p;
   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

Answer No. 04

```
#include<bits/stdc++.h>
using namespace std;
int prec(char c)
{
   if(c == '+' || c == '-')
      return 1;
   else if(c == '*' || c == '/')
      return 2;
   else
      return 0;
}
int main()
{
   string R;
   cin>>R;
   stack<char>st;
   string p;
   for(int i = 0; i < R.size(); i++)
   {
      if(R[i] >= '0' && R[i] <= '9')
         p+=R[i];
      else if(R[i] == '(')
         st.push('(');
      else if(R[i] == ')')
      {
         while(st.size()!=0 && st.top() != '(')
         {
            p+=st.top();
            st.pop();
```

```
        }
        if(st.top() == '(')
        {
            st.pop();
        }
    }
    else{
        while(st.size() && prec(R[i]) <= prec(st.top()))
        {
            p+=st.top();
            st.pop();
        }
        st.push(R[i]);
    }
}
while(st.size()!=0)
{
    p+= st.top();
    st.pop();
}
stack<int>S;
for(int i=0;i<p.size();i++)
{
    if(p[i]>='0'&&p[i]<='9')
    {
        S.push(p[i]-48);
    }
    else
    {
        if(p[i]=='+')
        {
            int a=S.top();S.pop();
            int b=S.top();S.pop();
            S.push(a+b);
        }
        if(p[i]=='-')
        {
            int a=S.top();S.pop();
            int b=S.top();S.pop();
            S.push(b-a);
        }
        if(p[i]=='*')
        {
```

```
            int a=S.top();S.pop();
            int b=S.top();S.pop();
            S.push(a*b);
        }
        if(p[i]=='/')
        {
            int a=S.top();S.pop();
            int b=S.top();S.pop();
            S.push(b/a);
        }
    }

  }
  cout<<S.top();
  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 wish>
class node
{
public:
   wish data;
   node *nxt;
   node *prv;
};
template<class wish>
class Deque
{
public:
   node<wish>*head;
   node<wish>*tail;
   int sz;
   Deque()
```

```
{
   head=NULL;
   tail=NULL;
   sz=0;
}
node<wish>*Create_New_Node(wish value)
{
   node<wish>*newnode=new node<wish>;
   newnode->data = value;
   newnode->nxt = NULL;
   newnode->prv = NULL;
   return newnode;
}
void push_back(wish value)
{
   node<wish>*newnode=Create_New_Node(value);
   if(sz==0)
   {
      head=newnode;
      tail=newnode;
      sz++;
      return;
   }
   tail->nxt=newnode;
   newnode->prv=tail;
   tail=newnode;
   sz++;
}
void push_front(wish value)
{
   node<wish>*newnode=Create_New_Node(value);
   if(sz==0)
   {
      head=newnode;
      tail=newnode;
      sz++;
      return;
   }
   sz++;
   head->prv=newnode;
   newnode->head;
   head=newnode;
}
```

```
void pop_back()
{

   if(sz==1)
   {
      delete tail;
      head=NULL;
      tail=NULL;
      sz--;
      return;
   }
   if(sz==0)
   {
      return;
   }
   node<wish>*a=tail;
   tail=tail->prv;
   delete a;
   tail->nxt=NULL;
   sz--;
}
void pop_front()
{
   if(sz==0)
   {
      return;
   }
   if(sz==1)
   {
      delete tail;
      head=NULL;
      tail=NULL;
      sz--;
      return;
   }
   node<wish>*a=head;
   head=head->nxt;
   delete a;
   head->prv=NULL;
   sz--;
}
int front()
{
```

```
        if(sz==0)
        {
            return -1;
        }
        return head->data;
    }
    int back()
    {
        if(sz==0)
        {
            return -1;
        }
        return tail->data;
    }
};
int main()
{
    Deque<int>dq;
    for(int i=1;i<=5;i++)
    {
        int a;cin>>a;
        dq.push_back(a);
    }
    cout<<dq.front()<<" ";
    cout<<dq.back()<<" ";
    dq.pop_back();dq.pop_front();
    cout<<dq.front()<<" ";
    cout<<dq.back()<<" ";
}
```

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 s;
   cin>>s;
   deque<char>dq;
   for(int i=0;i<s.size();i++)
   {
      dq.push_back(s[i]);
   }
   while(dq.size()!=0)
   {
      if(dq.front()!=dq.back())
      {
         cout<<"No\n";
         return 0;
      }
      else if(dq.size()==1)
      {
         dq.pop_back();
      }
      else
      {
         dq.pop_back();
         dq.pop_front();
      }
   }
   cout<<"Yes\n";
}
```

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 n)
{
   auto a=l.begin();
   int i=0;
   while(a!=l.end())
   {
      if(*a==n)
      {
         break;
      }
      a++;
      i++;
   }
   advance(a,i);
   l.erase(a);
   return;
}
int main()
{
   list<int>l;
   for(int i=1;i<=6;i++)
   {
      int a;cin>>a;
      l.push_back(a);
   }
   int n;
   cin>>n;
   deleteValue(l,n);
   auto a=l.begin();
   while(a!=l.end())
   {
      cout<<*a<<" ";
      a++;
   }
   cout<<"\n";
}
```