Experiment 4

Student Name: Shiv kant kumar **UID: 21BCS10055**
Branch: CSE **Section/Group:** 20BCS-MM-902/B
Semester: 5th **Date of Performance:** 03/09/2022 **Subject Name:** DAA Lab **Subject Code:** 21-CSP-312

1. Aim/Overview of the practical:

(i) Code to Insert and Delete an element at the beginning and at end in Doubly and Circular Linked List.

2. Task to be done/ Which logistics used:

Insert and delete an element from a doubly circular linked list.

3. Algorithm/Flowchart:

1. Start.
2. For insertion in the end if the list is empty start pointer points to the first node the list.If the list is non empty previous pointer of M points to last node, next pointer of M points to first node and last node's next pointer points to this M node and first node's previous pointer points to this M node
3. For Insertion at the beginning if the list is empty T next pointer points to first node of the list, T previous pointer points to last node the list, last node's next pointer points to this T node, first node's previous pointer also points this T node and shift 'Start' pointer to this T node.
4. If the list is not empty, then we define two pointers curr and prev_1 and initialize the pointer curr points to the first node of the list, and prev_1 = NULL.
5. Traverse the list using the curr pointer to find the node to be deleted and before moving from curr to the next node, every time set prev_1 = curr.
6. If the node is found, check if it is the only node in the list. If yes, set start = NULL and free the node pointing by curr.
7. If the list has more than one node, check if it is the first node of the list. The condition to check this is (curr == start). If yes, then move prev_1 to the last node(prev_1 = start -> prev).

8. If curr is not the first node, we check if it is the last node in the list. The condition to check this is (curr > next == start). If yes, set prev_1 -> next = start and start -> prev = prev_1. Free the node pointing by curr.
9. If the node to be deleted is neither the first node nor the last node, declare one more pointer temp and initialize the pointer temp points to the next of curr pointer (temp = curr>next). Now set, prev_1 -> next = temp and temp ->prev = prev_1. Free the node pointing by curr. 8.
10. Stop and print the result.

4. Steps for experiment/practical/Code:

```
#include<iostream>  using
namespace std;

class node { public:
            node* next;
            node*
            prev;      int
            data;
};
void insert_front(node** head)
{
         cout << "\nEnter Data to insert at front :\n";
         node* new_node = new node; cin
         >>  new_node->data;  if  (*head
         == NULL) { new_node->next =
         new_node;
      new_node->prev = new_node;
                 *head = new_node;
         } else { new_node->next =
*head;  new_node->prev = (*head)->prev; ((*head)-
      >prev)-
      >next = new_node;
                 (*head)->prev = new_node;
                 *head = new_node;
         }
         cout << "Data inserted at front\n";
}
void insert_end(node** head)
```

```
   { cout << "\nEnter Data to insert at end :\n";

              node* new_node = new node;
              cin >> new_node->data;

              if (*head == NULL) { new_node-
                 > next      =         new_node;
new_node->prev = new_node;   *head =
new_node; }    else {             node* curr =
*head;   while (curr->next != *head)
            curr = curr->next;
            new_node->next = curr->next;

                     new_node->prev = curr;
                     (curr->next)->prev = new_node;

                     curr->next = new_node;
              }


              cout << "Data inserted at last\n";
   }
   void delete_front(node** head)
   { if (*head == NULL) { cout << "\nList in
             empty!!\n";  }
              else if ((*head)->next == *head) {
          delete *head;
          *head = NULL;
              }        else {
                 node* curr = new node; curr = (*head)->next;
                     curr->prev =
        (*head)->prev;     ((*head)>prev)->next =
                                        curr; delete
                     *head;
                     *head = curr;
              }
              cout << "\nData Deleted from front\n";
   }
```

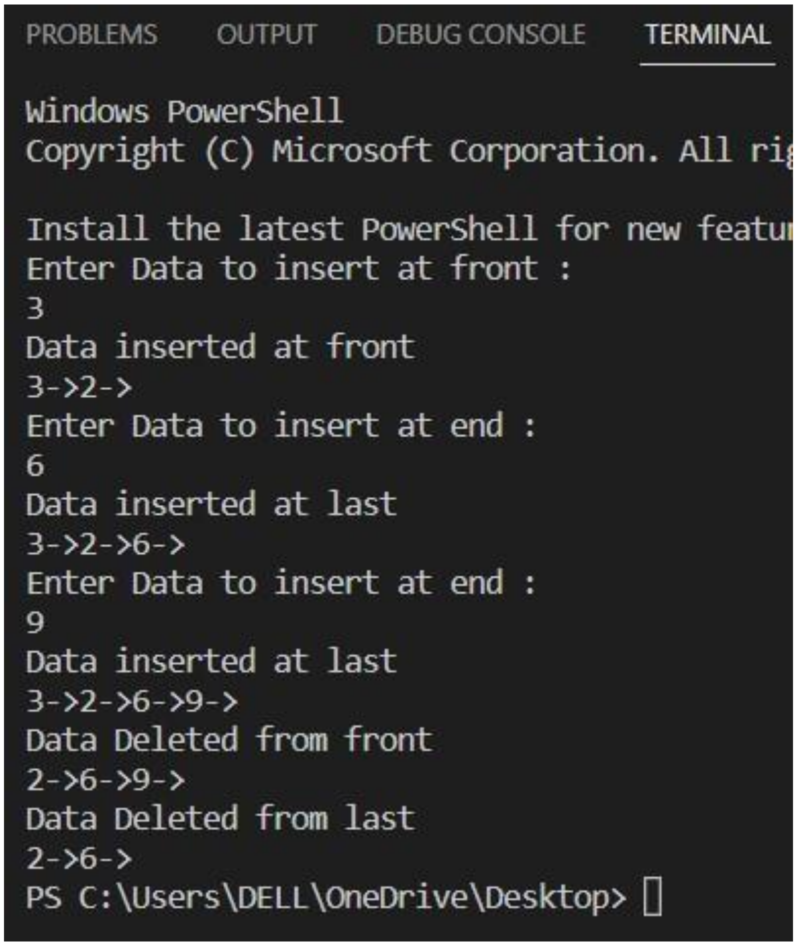
```
void delete_end(node** head)  { if (*head
== NULL) { cout << "\nList is
Empty!!\n";
          }
          else if ((*head)->next == *head) {
  delete *head; *head = NULL;
          }


          else {   node* curr = new node; curr
                    = *head;
                    while (curr->next != (*head)) {
                              curr = curr->next;
                    }
                              (curr->prev)->next = curr->next;
          (curr->next)->prev = curr->prev;
          delete curr;
          } cout << "\nData Deleted from last\n";
}
void display(node* head)
{ node* curr = head; if (curr == NULL)  cout << "\n List is Empty!!"; else { do { cout
          << curr->data << "->"; curr = curr->next;
                    } while (curr != head);
          }  }
int main()
{ int choice;
          char menu = 'y'; node* head
          = NULL;
          insert_front(&head);
    display(head);
    insert_front(&head);
    display(head);
    insert_end(&head);
    display(head);
    insert_end(&head);
    display(head);
    delete_front(&head);
    display(head);
    delete_end(&head);
    display(head); return 0;
}
```

5. Observations/Discussions/Complexity Analysis:

Time Complexity: O(n)

6. Result/Output/Writing Summary:



```
Windows PowerShell
Copyright (C) Microsoft Corporation. All ri

Install the latest PowerShell for new featu
Enter Data to insert at front :
3
Data inserted at front
3->2->
Enter Data to insert at end :
6
Data inserted at last
3->2->6->
Enter Data to insert at end :
9
Data inserted at last
3->2->6->9->
Data Deleted from front
2->6->9->
Data Deleted from last
2->6->
PS C:\Users\DELL\OneDrive\Desktop> 
```

1. Aim/Overview of the practical:

(ii) Code to push & pop and check Isempty, Isfull and Return top element in stacks using templates.

2. Task to be done/ Which logistics used:

Using C++ templates perform push and pop operation on stacks.

3. Algorithm/Flowchart:

1. Start.
2. First we will define the size.
3. Then we will create a class template called Stack.
4. Then we will check the top of stack using - template <class T> Stack<T>::Stack() { top = -1;
5. Then we will push elements into the stack using templates.
6. Using template, we will check whether the stack is empty or is full.
7. The we will pop an element of stack using templates.
8. We will check the top element using template <class T> T Stack<T>::topElement().
9. Print the result.
10. Stop.

4. Steps for experiment/practical/Code:

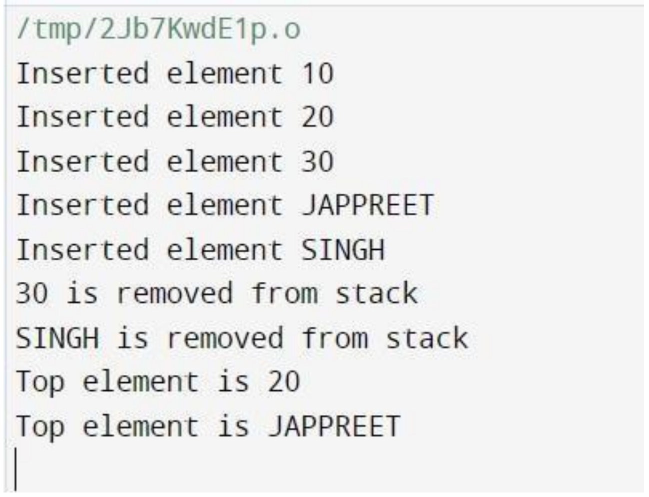
```
#include  <iostream>  #include  <string>
using  namespace  std;  #define  SIZE  5
template <class T> class Stack { public:
      Stack(); void
    push(T k);
      T pop();
      T topElement();
      bool  isFull(); bool
      isEmpty();
private:
      int top;
    T st[SIZE];
};
template <class T> Stack<T>::Stack() { top = -1; } template
<class T> void Stack<T>::push(T k)
{ if (isFull()) { cout << "Stack is
      full\n";
      }
      cout << "Inserted element " << k <<
      endl; top = top + 1; st[top] = k;
}
template <class T> bool Stack<T>::isEmpty()
{ if (top == -1) return 1; else
      return 0;
}
template <class T> bool Stack<T>::isFull()
{
      if (top == (SIZE - 1))          return 1; else
          return 0;
}


template <class T> T Stack<T>::pop()
{
      T popped_element = st[top]; top--; return
      popped_element;
}
template <class T> T Stack<T>::topElement()
{
      T top_element = st[top];
      return top_element;
} int main()
{
      Stack<int> integer_stack; Stack<string>
    string_stack; integer_stack.push(10);
    integer_stack.push(20);
    integer_stack.push(30);
    string_stack.push("JAPPREET");
    string_stack.push("SINGH");
      cout << integer_stack.pop() << " is removed from stack" << endl;
      cout << string_stack.pop() << " is removed from stack "  << endl; cout
      << "Top element is " << integer_stack.topElement()<< endl; cout
      << "Top element is " << string_stack.topElement()<< endl; return 0;  }
```

5. Observations/Discussions/ Complexity Analysis:

Time Complexity: O(1)

6. Result/Output/Writing Summary:



```
/tmp/2Jb7KwdE1p.o
Inserted element 10
Inserted element 20
Inserted element 30
Inserted element JAPPREET
Inserted element SINGH
30 is removed from stack
SINGH is removed from stack
Top element is 20
Top element is JAPPREET
```