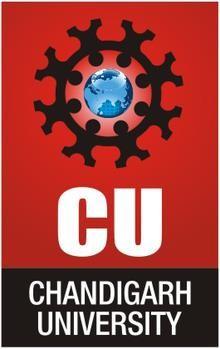
**CHANDIGARH UNIVERSITY**

UNIVERSITY INSTITUTE OF ENGINEERING

**DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING**



|  |  |
| --- | --- |
| **Submitted By:                                                                          Submitted To:**  Yash Gupta Monika(E12802) | |
| **Subject Name** | Design Analysis and Algorithm |
| **Subject Code** | 20CSP\_312 |
| **Branch** | CSE |
| **Semester** | 5th |

**LAB -INDEX**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Sr.No** | **Program** | **Date** | **Evaluation** | | | | **Sign** |
| **LW(12)** | **VV(8)** | **FW(10)** | **Total (30)** |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |

**Experiment 4**

**1. Aim/Overview of the practical:**

A - Code to Insert and Delete an element at the beginning and at end in Doubly and Circular Linked List

B - Code to push & pop and check Isempty, Isfull,and Return top element in stacks using templates.

**2. Algorithm:**

Step 1: IF PTR = NULL

Write OVERFLOW

Go to Step 11

[END OF IF]

Step 2: SET NEW\_NODE = PTR

Step 3: SET PTR = PTR -> NEXT

Step 4: SET NEW\_NODE -> DATA = VAL

Step 5: SET NEW\_NODE -> NEXT = NULL

Step 6: SET TEMP = START

Step 7: Repeat Step 8 while TEMP -> NEXT != NULL

Step 8: SET TEMP = TEMP -> NEXT

Step 9: SET TEMP -> NEXT = NEW\_NODE

Step 10: SET NEW\_NODE -> PREV = TEMP

Step 1 : IF ptr = NULL

Write OVERFLOW

Go to Step 9

Step 2 : SET NEW\_NODE = ptr

Step 3 : SET ptr = ptr -> NEXT

Step 4 : SET NEW\_NODE -> DATA = VAL

Step 5 : SET NEW\_NODE -> PREV = NULL

Step 6 : SET NEW\_NODE -> NEXT = START

Step 7 : SET head -> PREV = NEW\_NODE

Step 8 : SET head = NEW\_NODE

**3. Steps for experiment/practical/Code:**

A -

#include <bits/stdc++.h>

using namespace std;

class Node

{

public:

int data;

Node \*next;

Node \*prev;

};

void push(Node \*\*head\_ref, int new\_data)

{

Node \*new\_node = new Node();

new\_node->data = new\_data;

new\_node->next = (\*head\_ref);

new\_node->prev = NULL;

if ((\*head\_ref) != NULL)

(\*head\_ref)->prev = new\_node;

(\*head\_ref) = new\_node;

}

void insertAfter(Node \*prev\_node, int new\_data)

{

if (prev\_node == NULL)

{

cout << "the given previous node cannot be NULL";

return;

}

Node \*new\_node = new Node();

new\_node->data = new\_data;

new\_node->next = prev\_node->next;

prev\_node->next = new\_node;

new\_node->prev = prev\_node;

if (new\_node->next != NULL)

new\_node->next->prev = new\_node;

}

void append(Node \*\*head\_ref, int new\_data)

{

Node \*new\_node = new Node();

Node \*last = \*head\_ref;

new\_node->data = new\_data;

new\_node->next = NULL;

if (\*head\_ref == NULL)

{

new\_node->prev = NULL;

\*head\_ref = new\_node;

return;

}

while (last->next != NULL)

last = last->next;

last->next = new\_node;

new\_node->prev = last;

return;

}

void printList(Node \*node)

{

Node \*last;

cout << "\nTraversal in forward direction \n";

while (node != NULL)

{

cout << " " << node->data << " ";

last = node;

node = node->next;

}

cout << "\nTraversal in reverse direction \n";

while (last != NULL)

{

cout << " " << last->data << " ";

last = last->prev;

}

}

int main()

{

Node \*head = NULL;

append(&head, 6);

push(&head, 7);

push(&head, 1);

append(&head, 4);

insertAfter(head->next, 8);

cout << "Created DLL is: ";

printList(head);

return 0;

B-

#include <bits/stdc++.h>

using namespace std;

template <class X>

class st

{

    X \*arr;

    int top;

    int capacity;

public:

    st(int size = SIZE;);

    void push(X);

    X pop();

    X peek();

    int size();

    bool isEmpty();

    bool isFull();

    ~st()

    {

        delete[] arr;

    }

};

template <class X>

st<X>::st(int size)

{

    arr = new X[size];

    capacity = size;

    top = -1;

}

template <class X>

void st<X>::push(X x)

{

    if (isFull())

    {

        cout << "overflow" << endl;

        exit();

    }

    cout << "Inserting the value " << x << endl;

    arr[++top] = x;

}

template <class X>

X st<X>::pop()

{

    if (isEmpty())

    {

        cout << "underflow" << endl;

        exit();

    }

    cout << "removing the value " << peek() << endl;

    return arr[top--];

}

template <class X>

X st<X>::peek()

{

    if (!isEmpty())

        return arr[top];

    else

        exit(EXIT\_FAILURE);

}

template <class X>

int st<X>::size()

{

    return top + 1;

}

template <class X>

bool st<X>::isEmpty()

{

    return top == -1;

}

template <class X>

bool st<X>::isFull()

{

    return top == capacity - 1;

}

int main()

{

    st<string> pt(2);

    pt.push("A");

    pt.push("B");

    pt.pop();

    pt.pop();

    pt.push("C");

    // Prints the top of the stack

    cout << "Top element is: " << pt.peek() << endl;

    // Returns the number of elements present in the stack

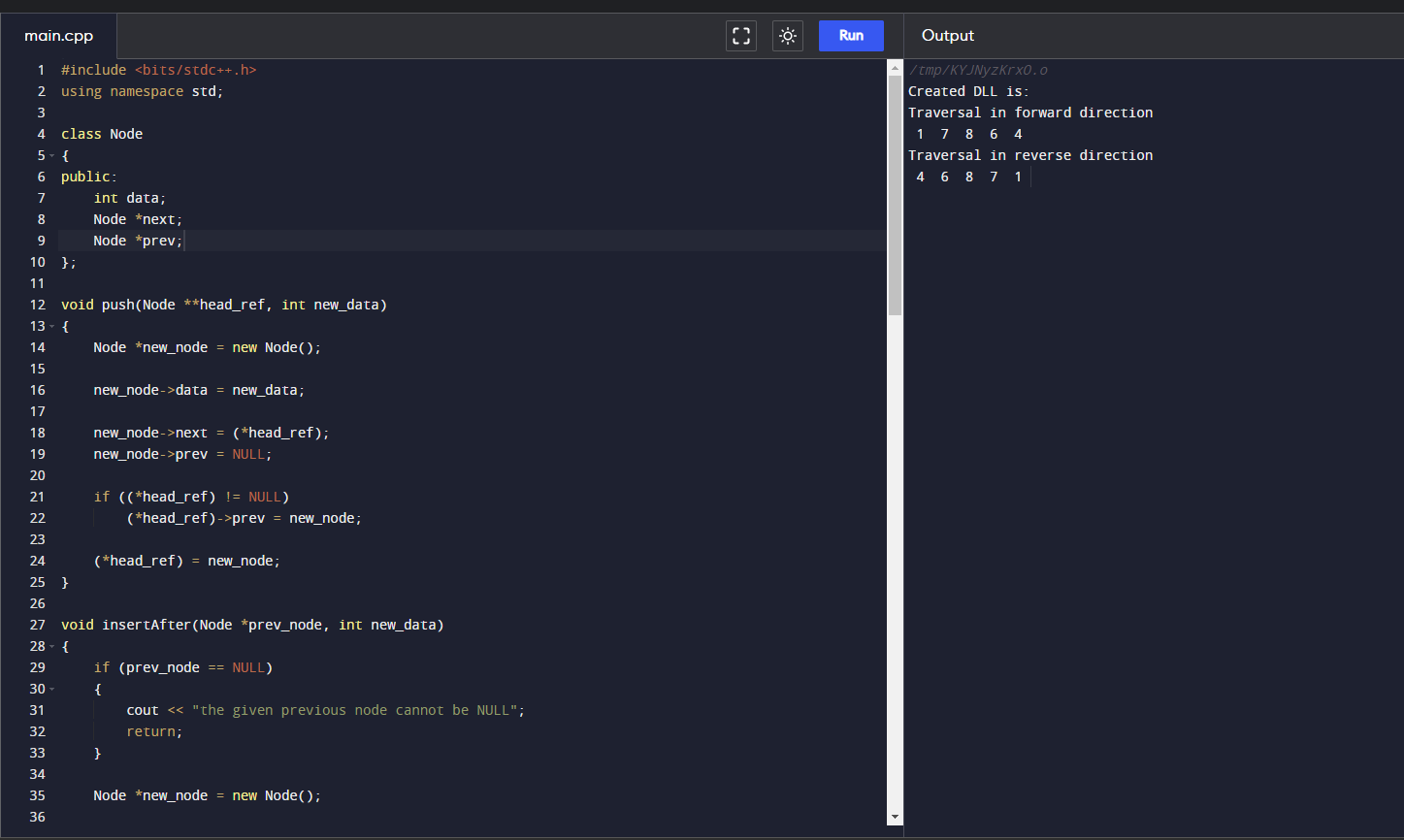
    cout << "Stack size is " << pt.size() << endl;

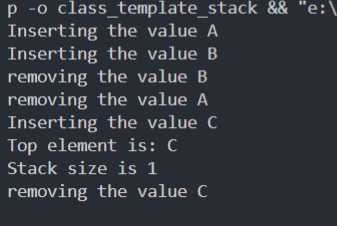
    pt.pop();

    return 0;

}

}**4. Result/Output/Writing Summary:**





**5. Observations/Discussions/ Complexity Analysis:**

Time complexity of finding frequency of elements of an array is O(n).

**6. Learning outcomes (What I have learnt):**

**1.** To learn how to calculate the frequency of the elements of an array.

**2.** To learn how to use for loop in these cases.

**Evaluation Grid (To be created as per the SOP and Assessment guidelines by the faculty):**

|  |  |  |  |
| --- | --- | --- | --- |
| Sr. No. | Parameters | Marks Obtained | Maximum Marks |
| 1. |  |  |  |
| 2. |  |  |  |
| 3. |  |  |  |
|  |  |  |  |