

Experiment-1

1. Perform Linear Search and Binary Search on an array.

Algorithm:

Source Code:

1.Linear Search:

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

int search(int arr[], int N, int x)
```

```

{
    int i;
    for (i = 0; i < N; i++)
        if (arr[i] == x)
            return i;
    return -1;
}

int main(void)
{
    int arr[] = { 2, 3, 4, 10, 40 };
    int x = 10;
    int N = sizeof(arr) / sizeof(arr[0]);

    int result = search(arr, N, x);
    (result == -1)
        ? cout << "Element is not present in array"
        : cout << "Element is present at index " << result;
    return 0;
}

```

Output:

```

PS C:\Users\Mohit> cd "c:\Code\" ; if ($?) { g++ tempCodeRunnerFile.cpp -o tempCodeRunnerFile } ; if ($?) { .\tempCodeRunnerFile }
Element is present at index 3
PS C:\Code>

```

2.Binary Search:

```

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

int binarySearch(int arr[], int size, int k)
{
    int start = 0;
    int end = size - 1;
    int mid = start + (end - start) / 2;

    while (start <= end)
    {
        if (arr[mid] == k)
        {
            return mid;
        }
        if (arr[mid] < k)

```

```

        {
            start = mid + 1;
        }
        else
        {
            end = mid - 1;
        }
        mid = start + (end - start) / 2;
    }
    return -1;
}

int main()
{
    int arr1[6] = {1, 5, 6, 7, 8, 9};
    int index = binarySearch(arr1, 6, 7);
    cout << "index of 7 is:" << index + 1 << endl;

    return 0;
}

```

Output:

```

PS C:\Code> cd "c:\Code\" ; if ($?) { g++ tempCodeRunnerFile.cpp -o tempCodeRunnerFile } ; if ($?) { .\tempCodeRunnerFile }
index of 7 is:4
PS C:\Code>

```

Learning Outcome:

Experiment-2

1. Create a stack and perform Pop, Push, and Traverse operations on the stack using arrays.

Algorithm:

Source Code:

```

#include<stdio.h>
int stack[100],choice,n,top,x,i;
void push(void);
void pop(void);
void display(void);
int main()
{
    //clrscr();
    top=-1;
    printf("\n Enter the size of STACK[MAX=100]:");
    scanf("%d",&n);
    printf("\n\t STACK OPERATIONS USING ARRAY");
    printf("\n\t-----");
    printf("\n\t 1.PUSH\n\t 2.POP\n\t 3.DISPLAY\n\t 4.EXIT");
    do
    {
        printf("\n Enter the Choice:");
        scanf("%d",&choice);
        switch(choice)
        {
            case 1:
            {
                push();
                break;
            }
            case 2:
            {
                pop();
                break;
            }
            case 3:
            {
                display();
                break;
            }
            case 4:
            {
                printf("\n\t EXIT POINT ");
                break;
            }
            default:
            {
                printf("\n\t Please Enter a Valid Choice(1/2/3/4)");
            }
        }
    }
    while(choice!=4);
}

```

```

    return 0;
}
void push()
{
    if(top>=n-1)
    {
        printf("\n\tSTACK is over flow");

    }
    else
    {
        printf(" Enter a value to be pushed:");
        scanf("%d",&x);
        top++;
        stack[top]=x;
    }
}
void pop()
{
    if(top<=-1)
    {
        printf("\n\t Stack is under flow");
    }
    else
    {
        printf("\n\t The popped elements is %d",stack[top]);
        top--;
    }
}
void display()
{
    if(top>=0)
    {
        printf("\n The elements in STACK \n");
        for(i=top; i>=0; i--)
            printf("\n%d",stack[i]);
        printf("\n Press Next Choice");
    }
    else
    {
        printf("\n The STACK is empty");
    }
}
}

```

Output:

```
PS C:\Users\Mohit> cd "c:\Code\" ; if ($?) { gcc file.c -o file } ; if ($?) { .\file }  
  
Enter the size of STACK[MAX=100]:5  
  
STACK OPERATIONS USING ARRAY  
  
Enter the Choice:3  
  
The elements in STACK  
  
3  
2  
1  
Press Next Choice  
Enter the Choice:2  
  
The popped elements is 3  
Enter the Choice:3  
  
The elements in STACK  
  
2  
1  
Press Next Choice  
Enter the Choice:4  
  
EXIT POINT  
PS C:\Code> █
```

Learning Outcome:

Experiment-3

1. Create a stack and perform Pop, Push, and Traverse operations on the stack using a Linear Linked list.

Algorithm:

Source Code:

```
#include <stdio.h>
#include <stdlib.h>
```



```

void push();
void pop();
void display();
struct node
{
int val;
struct node *next;
};
struct node *head;

void main ()
{
    int choice=0;
    printf("\n*****Stack operations using linked list*****\n");
    printf("\n-----\n");
    while(choice != 4)
    {
        printf("\n\nChose one from the below options...\n");
        printf("\n1.Push\n2.Pop\n3.Show\n4.Exit");
        printf("\n Enter your choice \n");
        scanf("%d",&choice);
        switch(choice)
        {
            case 1:
            {
                push();
                break;
            }
            case 2:
            {
                pop();
                break;
            }
            case 3:
            {
                display();
                break;
            }
            case 4:
            {
                printf("Exiting....");
                break;
            }
            default:
            {
                printf("Please Enter valid choice ");
            }
        }
    };
}

```

```

}
}
void push ()
{
    int val;
    struct node *ptr = (struct node*)malloc(sizeof(struct node));
    if(ptr == NULL)
    {
        printf("not able to push the element");
    }
    else
    {
        printf("Enter the value");
        scanf("%d",&val);
        if(head==NULL)
        {
            ptr->val = val;
            ptr -> next = NULL;
            head=ptr;
        }
        else
        {
            ptr->val = val;
            ptr->next = head;
            head=ptr;
        }
        printf("Item pushed");
    }
}

void pop()
{
    int item;
    struct node *ptr;
    if (head == NULL)
    {
        printf("Underflow");
    }
    else
    {
        item = head->val;
        ptr = head;
        head = head->next;
        free(ptr);
        printf("Item popped");
    }
}

```

```

    }
}
void display()
{
    int i;
    struct node *ptr;
    ptr=head;
    if(ptr == NULL)
    {
        printf("Stack is empty\n");
    }
    else
    {
        printf("Printing Stack elements \n");
        while(ptr!=NULL)
        {
            printf("%d\n",ptr->val);
            ptr = ptr->next;
        }
    }
}
}

```

Output:

```

PS C:\Users\Mohit> cd "c:\Code\" ; if ($?) { gcc file.c -o file } ; if ($?) { .\file }

*****Stack operations using linked list*****

-----

Chose one from the below options...

1.Push
2.Pop
3.Show
4.Exit
Enter your choice
1
Enter the value5
Item pushed

Chose one from the below options...

1.Push
2.Pop
3.Show
4.Exit
Enter your choice
1
Enter the value6
Item pushed

Chose one from the below options...

1.Push
2.Pop
3.Show
4.Exit
Enter your choice
3
Printing Stack elements
6
5

```

Chose one from the below options...

- 1.Push
- 2.Pop
- 3.Show
- 4.Exit

Enter your choice

2

Item popped

Chose one from the below options...

- 1.Push
- 2.Pop
- 3.Show
- 4.Exit

Enter your choice

3

Printing Stack elements

5

Chose one from the below options...

- 1.Push
- 2.Pop
- 3.Show
- 4.Exit

Enter your choice

4

Exiting....

PS C:\Code> █

Learning Outcome: