S.No: 1	Exp. Name: ***Write a C program to Sort given elements using Bubble sort***	**Date: 2023-04-20**

Aim:

Develop an algorithm, implement and execute a **C** program that reads **n** integer numbers and arrange them in **ascending order** using **Bubble Sort**.

Source Code:

Lab7.c

ID: 22B01A4609

2022-2026-CSE-CS-A

Shri Vishnu Engineering College for Women (Autonomous)

```
#include<stdio.h>
void bubble_sort();
void print_array();
void main()
{
        int i,n,a[20];
        scanf("%d",&n);
        printf("Elements: ");
        for(i=0;i<n;i++)
        {
                scanf("%d",&a[i]);
        }
        printf("Before sorting: ");
        for(i=0;i<n;i++)
        {
                printf("%d ",a[i]);
        }
        bubble_sort(a,n);
}
void bubble_sort(int a[],int n)
{
        int temp,i,j;
        for(i=0;i<n;i++)
        {
                for(j=0;j<n-i-1;j++)
                {
                        if(a[j]>a[j+1])
                        {
                                temp=a[j];
                                a[j]=a[j+1];
                                a[j+1]=temp;
                        }
                }
        }
        print_array(a,n);
}
void print_array(int a[],int n)
{
        int i;
        printf("\nAfter sorting: ");
        for(i=0;i<n;i++)
        {
                printf("%d ",a[i]);
        }
        printf("\n");

        }
```

Execution Results - All test cases have succeeded!

Test Case - 1
User Output
4

Elements:
44 22 66 11
Before sorting: 44 22 66 11
After sorting: 11 22 44 66

Test Case - 2
User Output
5
Elements:
9 2 7 1 6
Before sorting: 9 2 7 1 6
After sorting: 1 2 6 7 9

S.No: 2	Exp. Name: ***Write a C program to Sort given elements using Selection sort smallest element method***	Date: 2023-04-20

ID: 22B01A4609

2022-2026-CSE-CS-A

Shri Vishnu Engineering College for Women (Autonomous)

Aim:

Write a program to `sort` (`Ascending order`) the given array elements using `selection sort smallest element` method.

At the time of execution, the program should print the message on the console as:

```
Enter value of n :
```

For example, if the user gives the **input** as:

```
Enter value of n : 3
```

Next, the program should print the messages one by one on the console as:

```
Enter element for a[0] :
Enter element for a[1] :
Enter element for a[2] :
```

if the user gives the **input** as:

```
Enter element for a[0] : 22
Enter element for a[1] : 33
Enter element for a[2] : 12
```

then the program should **print** the result as:

```
Before sorting the elements in the array are
Value of a[0] = 22
Value of a[1] = 33
Value of a[2] = 12
After sorting the elements in the array are
Value of a[0] = 12
Value of a[1] = 22
Value of a[2] = 33
```

Note: Do use the **printf()** function with a **newline** character (`\n`) at the end.

Source Code:

Program507.c

```c
#include<stdio.h>
void selection_sort();
void display();
void main()
{
        int a[20],i,size;
        printf("Enter value of n : ");
        scanf("%d",&size);
        for(i=0;i<size;i++)
        {
                printf("Enter element for a[%d] : ",i);
                scanf("%d",&a[i]);
        }
        printf("Before sorting the elements in the array are\n");
        for(i=0;i<size;i++)
        {
                printf("Value of a[%d] = %d",i,a[i]);
                printf("\n");
        }
        selection_sort(a,size);
        printf("After sorting the elements in the array are\n");
        display(a,size);
}
void selection_sort(int a[],int size)
{
        int i,j,min,temp;
        for (i=0;i<size-1;i++)
        {
                min=i;
                for(j=i+1;j<size;j++)
                {
                        if(a[min]>a[j])
                        {
                                min=j;
                        }
                }
                if(min!=i)
                {
                        temp=a[min];
                        a[min]=a[i];
                        a[i]=temp;
                }
        }
}
void display(int a[],int size)
{
        int i;
        for(i=0;i<size;i++)
        {
                printf("Value of a[%d] = %d",i,a[i]);
                printf("\n");
        }
}
```

Execution Results - All test cases have succeeded!

Test Case - 1
User Output
Enter value of n :
5
Enter element for a[0] :
34
Enter element for a[1] :
78
Enter element for a[2] :
56
Enter element for a[3] :
12
Enter element for a[4] :
48
Before sorting the elements in the array are
Value of a[0] = 34
Value of a[1] = 78
Value of a[2] = 56
Value of a[3] = 12
Value of a[4] = 48
After sorting the elements in the array are
Value of a[0] = 12
Value of a[1] = 34
Value of a[2] = 48
Value of a[3] = 56
Value of a[4] = 78

ID: 22B01A4609
2022-2026-CSE-CS-A
Shri Vishnu Engineering College for Women (Autonomous)

S.No: 3	Exp. Name: ***Write a C program to Sort elements using Insertion Sort***	Date: 2023-05-04

ID: 22B01A4609

Shri Vishnu Engineering College for Women (Autonomous)

Aim:

Write a program to `sort` the elements in ascending order with `insertion sort` technique using **functions**.

At the time of execution, the program should print the message on the console as:

```
Enter n value :
```

For example, if the user gives the **input** as:

```
Enter n value : 3
```

Next, the program should print the message on the console as:

```
Enter 3 elements :
```

if the user gives the **input** as:

```
Enter 3 elements : 45 67 34
```

then the program should **print** the result as:

```
Elements before sorting : 45 67 34
Elements after sorting : 34 45 67
```

Note: Do use **printf()** with '`\n`' at the end of output.

Source Code:

FunctionsWithArrays5.c

```
#include <stdio.h>
void insertion_sort(int [], int);
void read(int [], int);
void display(int [], int);
void main() {
        int a[20], n, i;
        printf("Enter n value : ");
        scanf("%d", &n);
        read(a, n);
        printf("Elements before sorting : ");
        display(a, n);
        insertion_sort(a, n);
        printf("Elements after sorting : ");
        display(a, n);
        printf("\n");
}
void insertion_sort(int a[],int n)
{
        int i,j,key;
        for(i=1;i<n;i++)
        {
                key=a[i];
                j=i-1;
                while(j>=0&&a[j]>key)
                {
                        a[j+1]=a[j];
                        j=j-1;
                }
                a[j+1]=key;
        }// Write the arguments
        // Write the code
}
void read(int a[],int n)
{
        int i;
        printf("Enter %d elements : ",n);
        for(i=0;i<n;i++)
        {
                scanf("%d",&a[i]);
        }// Write the arguments
        // Write the code
}
void display(int a[],int n)
{
        int i;
        for(i=0;i<n;i++)
        {
                printf("%d ",a[i]);
        }// Write the arguments
        printf("\n");
        // Write the code
}
```

Execution Results - All test cases have succeeded!

Test Case - 1
User Output
Enter n value :
3
Enter 3 elements :
45 67 34
Elements before sorting : 45 67 34
Elements after sorting : 34 45 67

Test Case - 2
User Output
Enter n value :
5
Enter 5 elements :
2 8 4 1 3
Elements before sorting : 2 8 4 1 3
Elements after sorting : 1 2 3 4 8

Test Case - 3
User Output
Enter n value :
6
Enter 6 elements :
23 15 18 12 16 11
Elements before sorting : 23 15 18 12 16 11
Elements after sorting : 11 12 15 16 18 23

Test Case - 4
User Output
Enter n value :
4
Enter 4 elements :
-26 -19 -263 -189
Elements before sorting : -26 -19 -263 -189
Elements after sorting : -263 -189 -26 -19

Test Case - 5
User Output
Enter n value :
6
Enter 6 elements :
-2 -4 2 4 5 1
Elements before sorting : -2 -4 2 4 5 1

```
Elements after sorting : -4 -2 1 2 4 5
```

S.No: 4	Exp. Name: ***Write a C program to Sort given elements using Quick sort***	**Date: 2023-04-20**

ID: 22B01A4609

Shri Vishnu Engineering College for Women (Autonomous)

Aim:

Write a program to `sort` (`Ascending order`) the given elements using `quick sort` technique.

Note: Pick the last element as pivot. You will not be awarded marks if you do not follow this instruction.

At the time of execution, the program should print the message on the console as:

```
Enter array size :
```

For example, if the user gives the **input** as:

```
Enter array size : 5
```

Next, the program should print the following message on the console as:

```
Enter 5 elements :
```

if the user gives the **input** as:

```
Enter 5 elements : 34 67 12 45 22
```

then the program should **print** the result as:

```
Before sorting the elements are : 34 67 12 45 22
After sorting the elements are : 12 22 34 45 67
```

Note: Do use the **printf()** function with a **newline** character (`\n`).

Source Code:

QuickSortMain.c

```
#include <stdio.h>
#include "QuickSortFunctions.c"
void main() {
        int arr[15], i, n;
        printf("Enter array size : ");
        scanf("%d", &n);
        printf("Enter %d elements : ", n);
        for (i = 0; i < n; i++) {
                scanf("%d", &arr[i]);
        }
        printf("Before sorting the elements are : ");
        display(arr, n);
        quickSort(arr, 0, n - 1);
        printf("After sorting the elements are : ");
        display(arr, n);
}
```

QuickSortFunctions.c

```
void display(int arr[15], int n) {
        int i;
        for(i=0;i<n;i++)
        {
                printf("%d ",arr[i]);
        }
        printf("\n");

}
int partition(int arr[15], int lb, int ub) {
  quickSort(arr,lb,ub);
quickSort(arr,lb,ub);
}
void quickSort(int arr[15], int low, int high) {
        int i,j,pivot,temp;
        i=low;
        j=high;
        pivot=high;
if (low<high)
 {
        while(i<j)
        {
                while(arr[i]<arr[pivot])
                {
                        i=i+1;
                }
                while(arr[j]>arr[pivot])
                {
                        j=j-1;
                }
                if(j>i)
                {
         temp=arr[i];
         arr[i]=arr[j];
         arr[j]=temp;
        }
        }
        temp= arr[j];
        arr[j]=arr[pivot];
        arr[pivot]=temp;
        partition(arr,low,j-1);
        partition(arr,j+1,high);
 }
}
```

Execution Results - All test cases have succeeded!

Test Case - 1
User Output
Enter array size :
5
Enter 5 elements :

34 67 12 45 22
Before sorting the elements are : 34 67 12 45 22
After sorting the elements are : 12 22 34 45 67

Test Case - 2
User Output
Enter array size :
8
Enter 8 elements :
77 55 22 44 99 33 11 66
Before sorting the elements are : 77 55 22 44 99 33 11 66
After sorting the elements are : 11 22 33 44 55 66 77 99

Test Case - 3
User Output
Enter array size :
5
Enter 5 elements :
-32 -45 -67 -46 -14
Before sorting the elements are : -32 -45 -67 -46 -14
After sorting the elements are : -67 -46 -45 -32 -14

S.No: 5	Exp. Name: ***Write a C program to Sort given elements using Merge sort***	**Date: 2023-05-04**

ID: 22B01A4609

Shri Vishnu Engineering College for Women (Autonomous)

Aim:

Write a program to `sort` (`Ascending order`) the given elements using `merge sort` technique.

At the time of execution, the program should print the message on the console as:

```
Enter array size :
```

For example, if the user gives the **input** as:

```
Enter array size : 5
```

Next, the program should print the following message on the console as:

```
Enter 5 elements :
```

if the user gives the **input** as:

```
Enter 5 elements : 34 67 12 45 22
```

then the program should **print** the result as:

```
Before sorting the elements are : 34 67 12 45 22
After sorting the elements are : 12 22 34 45 67
```

Note: Do use the **printf()** function with a **newline** character (`\n`).

Source Code:

MergeSortMain.c

```
#include <stdio.h>
#include "MergeSortFunctions.c"
void main() {
        int arr[15], i, n;
        printf("Enter array size : ");
        scanf("%d", &n);
        printf("Enter %d elements : ", n);
        for (i = 0; i < n; i++) {
                scanf("%d", &arr[i]);
        }
        printf("Before sorting the elements are : ");
        display(arr, n);
        splitAndMerge(arr, 0, n - 1);
        printf("After sorting the elements are : ");
        display(arr, n);
}
```

MergeSortFunctions.c

```
void display(int arr[15], int n)
{
        int i;
        for(i=0;i<n;i++)
        {
          printf("%d ",arr[i]);
        }
        printf("\n");
}
void merge(int arr[15], int low, int mid, int high)
{
        int i,j,k,temp[15];
        i=low;
        j=mid+1;
        k=low;
        while(i<=mid&&j<=high)
        {
                if(arr[i]<arr[j])
                {
                        temp[k]=arr[i];
                        k++;
                        i++;
                }
                else
                {
                        temp[k]=arr[j];
                        k++;
                        j++;
                }
        }
        if(i>mid)
        {
                while(j<=high)
                {
                        temp[k]=arr[j];
                        k++;
                        j++;
                }
        }
        else
        {
                while(i<=mid)
                {
                        temp[k]=arr[i];
                    k++;
                        i++;
                }
        }
        for(i=low;i<=high;i++)
        {
                arr[i]=temp[i];
        }
}
void splitAndMerge(int arr[15], int low, int high)
{
```

```
        {
                mid=(low+high)/2;
                splitAndMerge(arr,low,mid);
                splitAndMerge(arr,mid+1,high);
                merge(arr,low,mid,high);
        }

}
```

Execution Results - All test cases have succeeded!

Test Case - 1
User Output
Enter array size :
5
Enter 5 elements :
34 67 12 45 22
Before sorting the elements are : 34 67 12 45 22
After sorting the elements are : 12 22 34 45 67

Test Case - 2
User Output
Enter array size :
8
Enter 8 elements :
77 55 22 44 99 33 11 66
Before sorting the elements are : 77 55 22 44 99 33 11 66
After sorting the elements are : 11 22 33 44 55 66 77 99

Test Case - 3
User Output
Enter array size :
5
Enter 5 elements :
-32 -45 -67 -46 -14
Before sorting the elements are : -32 -45 -67 -46 -14
After sorting the elements are : -67 -46 -45 -32 -14

S.No: 6	Exp. Name: ***Write a C program to implement different Operations on Stack using Array representation***	**Date: 2023-05-04**

Aim:

Write a program to implement `stack` using **arrays**.

```
Sample Input and Output:
    1.Push 2.Pop 3.Display 4.Is Empty 5.Peek 6.Exit
    Enter your option : 4
    Stack is empty.
    1.Push 2.Pop 3.Display 4.Is Empty 5.Peek 6.Exit
    Enter your option : 2
    Stack is underflow.
    1.Push 2.Pop 3.Display 4.Is Empty 5.Peek 6.Exit
    Enter your option : 3
    Stack is empty.
    1.Push 2.Pop 3.Display 4.Is Empty 5.Peek 6.Exit
    Enter your option : 5
    Stack is underflow.
    1.Push 2.Pop 3.Display 4.Is Empty 5.Peek 6.Exit
    Enter your option : 1
    Enter element : 25
    Successfully pushed.
    1.Push 2.Pop 3.Display 4.Is Empty 5.Peek 6.Exit
    Enter your option : 1
    Enter element : 26
    Successfully pushed.
    1.Push 2.Pop 3.Display 4.Is Empty 5.Peek 6.Exit
    Enter your option : 3
    Elements of the stack are : 26 25
    1.Push 2.Pop 3.Display 4.Is Empty 5.Peek 6.Exit
    Enter your option : 2
    Popped value = 26
    1.Push 2.Pop 3.Display 4.Is Empty 5.Peek 6.Exit
    Enter your option : 4
    Stack is not empty.
    1.Push 2.Pop 3.Display 4.Is Empty 5.Peek 6.Exit
    Enter your option : 5
    Peek value = 25
    1.Push 2.Pop 3.Display 4.Is Empty 5.Peek 6.Exit
    Enter your option : 6
```

Source Code:

StackUsingArray.c

ID: 22B01A4609

Shri Vishnu Engineering College for Women (Autonomous)

```c
#include <stdio.h>
#include <stdlib.h>
#define STACK_MAX_SIZE 10
#include "StackOperations.c"

int main() {
        int op, x;
        while(1) {
                printf("1.Push 2.Pop 3.Display 4.Is Empty 5.Peek 6.Exit\n");
                printf("Enter your option : ");
                scanf("%d", &op);
                switch(op) {
                        case 1:
                                printf("Enter element : ");
                                scanf("%d", &x);
                                push(x);
                                break;
                        case 2:
                                pop();
                                break;
                        case 3:
                                display();
                                break;
                        case 4:
                                isEmpty();
                                break;
                        case 5:
                                peek();
                                break;
                        case 6:
                                exit(0);
                }
        }
}
```

StackOperations.c

```
int arr[STACK_MAX_SIZE],top=-1;
void push(int element)
{
        if(top==STACK_MAX_SIZE-1)
        printf("Stack is overflow.");
        else{
                top++;
                arr[top]=element;
                printf("Successfully pushed.");
        }
        printf("\n");
}
void pop(){
        if(top==-1)
        printf("Stack is underflow.");
        else{
                int x=arr[top];
                top=top-1;
                printf("Popped value = %d",x);
        }
        printf("\n");
}
void display()
{
        if(top==-1)
printf("Stack is empty.");
else{
        printf("Elements of the stack are : ");
        for(int i=top;i>=0;i--)
        printf("%d ",arr[i]);
}
printf("\n");
}
void isEmpty(){
        if(top==-1)
        printf("Stack is empty.");
        else
        printf("Stack is not empty.");
        printf("\n");
}
void peek(){
        if(top==-1)
        printf("Stack is underflow.");
        else{
                printf("Peek value = %d",arr[top]);
        }
        printf("\n");
}
```


ID: 22B01A4609

Shri Vishnu Engineering College for Women (Autonomous)

Execution Results - All test cases have succeeded!

Test Case - 1
User Output

```
1.Push 2.Pop 3.Display 4.Is Empty 5.Peek 6.Exit
Enter your option :
1
Enter element :
10
Successfully pushed.
1.Push 2.Pop 3.Display 4.Is Empty 5.Peek 6.Exit
Enter your option :
1
Enter element :
20
Successfully pushed.
1.Push 2.Pop 3.Display 4.Is Empty 5.Peek 6.Exit
Enter your option :
1
Enter element :
30
Successfully pushed.
1.Push 2.Pop 3.Display 4.Is Empty 5.Peek 6.Exit
Enter your option :
3
Elements of the stack are : 30 20 10
1.Push 2.Pop 3.Display 4.Is Empty 5.Peek 6.Exit
Enter your option :
5
Peek value = 30
1.Push 2.Pop 3.Display 4.Is Empty 5.Peek 6.Exit
Enter your option :
2
Popped value = 30
1.Push 2.Pop 3.Display 4.Is Empty 5.Peek 6.Exit
Enter your option :
2
Popped value = 20
1.Push 2.Pop 3.Display 4.Is Empty 5.Peek 6.Exit
Enter your option :
3
Elements of the stack are : 10
1.Push 2.Pop 3.Display 4.Is Empty 5.Peek 6.Exit
Enter your option :
5
Peek value = 10
1.Push 2.Pop 3.Display 4.Is Empty 5.Peek 6.Exit
Enter your option :
4
Stack is not empty.
1.Push 2.Pop 3.Display 4.Is Empty 5.Peek 6.Exit
Enter your option :
2
Popped value = 10
1.Push 2.Pop 3.Display 4.Is Empty 5.Peek 6.Exit
```

Enter your option :
3
Stack is empty.
1.Push 2.Pop 3.Display 4.Is Empty 5.Peek 6.Exit
Enter your option :
4
Stack is empty.
1.Push 2.Pop 3.Display 4.Is Empty 5.Peek 6.Exit
Enter your option :
6

S.No: 7	Exp. Name: ***Write a C program to Convert an Infix expression into Postfix expression***	**Date: 2023-05-30**

Aim:

Write the code in the functions `convertInfix(char *e)`, `priority(char x)`, `push(char x)`, `pop()` and `isEmpty` in the below code according to the hints given as comment lines.

```
Sample Output -1
Enter the expression : A+B*(C+D)
Postfix expression : ABCD+*+
Sample Output -2
Enter the expression : A+D*C+E-F&D
Invalid symbols in infix expression. Only alphanumeric and { '+', '-','*', '%', '/' }
are allowed.
Sample Output -3
Enter the expression : A+B*C+(D*E
Invalid infix expression : unbalanced parenthesis.
```

Source Code:

Infix2PostfixMain.c

```
#include "Infix2PostfixOperation.c"
int main() {
        char exp[20];
        char *e, x;
        printf("Enter the expression : ");
        scanf("%s",exp);
        e = exp;
        convertInfix(e);
}
```

Infix2PostfixOperation.c

```
#include <stdlib.h>
#include <string.h>
#include <stdio.h>
#include <ctype.h>
#define STACK_MAX_SIZE 20
char stack [STACK_MAX_SIZE];


int top = -1;

//Return 1 if stack is empty else return 0.
int isEmpty() {
        if (top == -1)
        return 1;
        else
        return 0;


}

//Push the character into stack
void push(char x) {
top++;
        stack[top] = x;
}

//pop a character from stack
char pop() {
        char x = stack[top];
        top--;
        return x;

}

// Return 0 if char is '('
// Return 1 if char is '+' or '-'
// Return 2 if char is '*' or '/' or '%'
int priority(char x) {
        switch(x)
        {
                case'(': return 0;
                case'+':
                case'-': return 1;
                case'*':
                case'/':
                case'%': return 2;
        }
}

//Output Format
//if expression is correct then output will be Postfix Expression : <postfix notation>
//If expression contains invalid operators then output will be "Invalid symbols in infix
expression. Only alphanumberic and { '+', '-','*', '%%', '/' } are allowed."
//If the expression contains unbalanced paranthesis the output will be "Invalid infix
expression : unbalanced parenthesis."
```

```
        char postfix[20];
        char ch;
        while(e[i]!='\0')
        {
                ch = e[i];
                if(isalnum(ch))
                postfix[j++] = ch;
                else if(ch == '(')
                push(ch);
                else if(ch == ')')
                {
                        while(top>-1 && stack[top]!='(')
                        postfix[j++] = pop();
                        if(top>-1 && stack[top]!='(')
                        {
                                flag = 1;
                                break;
                        }
                        else
                        top--;
                }
                else
                {
                        if(ch == '+' || ch =='-' ||ch =='*' ||ch =='/')
                        {
                                while(top>-1 & priority(stack[top])>=priority(ch))
                                postfix[j++] = pop();
                                push(ch);
                        }
                        else
                        {
                                printf("Invalid symbols in infix expression. Only
alphanumeric and { '+', '-','*', '%%', '/' } are allowed.");
                                flag = 1;
                                break;
                        }
                }
i++;}
        while(top>-1 && flag == 0)
        {
                if(stack[top] == '(')
                {
                        printf("Invalid infix expression : unbalanced parenthesis.");
                        break;
                }
        postfix[j++] = pop();
        }
        postfix[j] = '\0';
            if(top == -1 && flag == 0)
            {
                printf("Postfix expression : ");
                printf("%s",postfix);
            }
            printf("\n");
}
```

Execution Results - All test cases have succeeded!

Test Case - 1
User Output
Enter the expression :
A+B*(C-D)
Postfix expression : ABCD-*+

Test Case - 2
User Output
Enter the expression :
A+B*C
Postfix expression : ABC*+

Test Case - 3
User Output
Enter the expression :
A+B*(C+D)*E+(F*G
Invalid infix expression : unbalanced parenthesis.

Test Case - 4
User Output
Enter the expression :
A+B*(S+W&L)
Invalid symbols in infix expression. Only alphanumeric and { '+', '-','*', '%', '/' } are allowed.

S.No: 8	Exp. Name: ***Write a C program to implement different Operations on Queue using Array representation***	**Date: 2023-05-30**

Aim:

Write a program to implement queue using **arrays**.

```
Sample Input and Output:
    1.Enqueue 2.Dequeue 3.Display 4.Is Empty 5.Size 6.Exit
    Enter your option : 1
    Enter element : 23
    Successfully inserted.
    1.Enqueue 2.Dequeue 3.Display 4.Is Empty 5.Size 6.Exit
    Enter your option : 1
    Enter element : 56
    Successfully inserted.
    1.Enqueue 2.Dequeue 3.Display 4.Is Empty 5.Size 6.Exit
    Enter your option : 3
    Elements in the queue : 23 56
    1.Enqueue 2.Dequeue 3.Display 4.Is Empty 5.Size 6.Exit
    Enter your option : 4
    Queue is not empty.
    1.Enqueue 2.Dequeue 3.Display 4.Is Empty 5.Size 6.Exit
    Enter your option : 5
    Queue size : 2
    1.Enqueue 2.Dequeue 3.Display 4.Is Empty 5.Size 6.Exit
    Enter your option : 2
    Deleted element = 23
    1.Enqueue 2.Dequeue 3.Display 4.Is Empty 5.Size 6.Exit
    Enter your option : 2
    Deleted element = 56
    1.Enqueue 2.Dequeue 3.Display 4.Is Empty 5.Size 6.Exit
    Enter your option : 4
    Queue is empty.
    1.Enqueue 2.Dequeue 3.Display 4.Is Empty 5.Size 6.Exit
    Enter your option : 6
```

Source Code:

QueueUsingArray.c

ID: 22B01A4609

Shri Vishnu Engineering College for Women (Autonomous)

```c
#include <conio.h>
#include <stdio.h>
#include "QueueOperations.c"
int main() {
        int op, x;
        while(1) {
                printf("1.Enqueue 2.Dequeue 3.Display 4.Is Empty 5.Size 6.Exit\n");
                printf("Enter your option : ");
                scanf("%d",&op);
                switch(op) {
                        case 1:
                                printf("Enter element : ");
                                scanf("%d",&x);
                                enqueue(x);
                                break;
                        case 2:
                                dequeue();
                                break;
                        case 3:
                                display();
                                break;
                        case 4:
                                isEmpty();
                                break;
                        case 5:
                                size();
                                break;
                        case 6: exit(0);
                }
        }
}
```

QueueOperations.c

```
#define MAX 50
int front = -1,rear = -1;
int queue_arr[MAX];
void enqueue(int x)
{
        if(rear == MAX-1)
        printf("Queue is overflow.\n");
        else
        {
                if(front == -1)
                front = 0;
                rear += 1;
                queue_arr[rear] = x;
                printf("Successfully inserted.\n");
        }
}
void dequeue()
{
        if(rear == -1 || front > rear)
        {
                printf("Queue is underflow.\n");
                return;
        }
        else
        {
                printf("Deleted element = %d\n",queue_arr[front]);
                front += 1;
        }
}
void display()
{
        int i;
        if(rear == -1 || front > rear)
        printf("Queue is empty.\n");
        else
        {
                printf("Elements in the queue : ");
                for(i=front;i<=rear;i++)
                printf("%d ",queue_arr[i]);
                printf("\n");
        }
}
void isEmpty()
{
        if(front == -1 || front > rear)
        printf("Queue is empty.\n");
        else
        printf("Queue is not empty.\n");
}
void size()
{
        if(front == -1)
        printf("Queue size : 0\n");
        else
        printf("Queue size : %d\n",rear - front + 1);
}
```

Execution Results - All test cases have succeeded!

Test Case - 1
User Output
1.Enqueue 2.Dequeue 3.Display 4.Is Empty 5.Size 6.Exit
Enter your option :
2
Queue is underflow.
1.Enqueue 2.Dequeue 3.Display 4.Is Empty 5.Size 6.Exit
Enter your option :
3
Queue is empty.
1.Enqueue 2.Dequeue 3.Display 4.Is Empty 5.Size 6.Exit
Enter your option :
4
Queue is empty.
1.Enqueue 2.Dequeue 3.Display 4.Is Empty 5.Size 6.Exit
Enter your option :
5
Queue size : 0
1.Enqueue 2.Dequeue 3.Display 4.Is Empty 5.Size 6.Exit
Enter your option :
1
Enter element :
14
Successfully inserted.
1.Enqueue 2.Dequeue 3.Display 4.Is Empty 5.Size 6.Exit
Enter your option :
1
Enter element :
78
Successfully inserted.
1.Enqueue 2.Dequeue 3.Display 4.Is Empty 5.Size 6.Exit
Enter your option :
1
Enter element :
53
Successfully inserted.
1.Enqueue 2.Dequeue 3.Display 4.Is Empty 5.Size 6.Exit
Enter your option :
3
Elements in the queue : 14 78 53
1.Enqueue 2.Dequeue 3.Display 4.Is Empty 5.Size 6.Exit
Enter your option :
5
Queue size : 3
1.Enqueue 2.Dequeue 3.Display 4.Is Empty 5.Size 6.Exit
Enter your option :
6

Test Case - 2
User Output
1.Enqueue 2.Dequeue 3.Display 4.Is Empty 5.Size 6.Exit
Enter your option :
1
Enter element :
25
Successfully inserted.
1.Enqueue 2.Dequeue 3.Display 4.Is Empty 5.Size 6.Exit
Enter your option :
2
Deleted element = 25
1.Enqueue 2.Dequeue 3.Display 4.Is Empty 5.Size 6.Exit
Enter your option :
2
Queue is underflow.
1.Enqueue 2.Dequeue 3.Display 4.Is Empty 5.Size 6.Exit
Enter your option :
3
Queue is empty.
1.Enqueue 2.Dequeue 3.Display 4.Is Empty 5.Size 6.Exit
Enter your option :
1
Enter element :
65
Successfully inserted.
1.Enqueue 2.Dequeue 3.Display 4.Is Empty 5.Size 6.Exit
Enter your option :
3
Elements in the queue : 65
1.Enqueue 2.Dequeue 3.Display 4.Is Empty 5.Size 6.Exit
Enter your option :
4
Queue is not empty.
1.Enqueue 2.Dequeue 3.Display 4.Is Empty 5.Size 6.Exit
Enter your option :
2
Deleted element = 65
1.Enqueue 2.Dequeue 3.Display 4.Is Empty 5.Size 6.Exit
Enter your option :
4
Queue is empty.
1.Enqueue 2.Dequeue 3.Display 4.Is Empty 5.Size 6.Exit
Enter your option :
5
Queue size : 0
1.Enqueue 2.Dequeue 3.Display 4.Is Empty 5.Size 6.Exit
Enter your option :
1

Enter element :
63
Successfully inserted.
1.Enqueue 2.Dequeue 3.Display 4.Is Empty 5.Size 6.Exit
Enter your option :
5
Queue size : 1
1.Enqueue 2.Dequeue 3.Display 4.Is Empty 5.Size 6.Exit
Enter your option :
6

S.No: 9	Exp. Name: ***Implementation of Circular Queue using Arrays***	Date: 2023-05-31

Aim:

Write a program to implement circular queue using **arrays**.

Note: Define the **MAX** value as 5.

Source Code:

CQueueUsingArray.c

```
#include <stdio.h>
#include <stdlib.h>
#include "CQueueOperations.c"

int main() {
        int op, x;
        while(1) {
                printf("1.Enqueue 2.Dequeue 3.Display 4.Is empty 5.Size 6.Exit\n");
                printf("Enter your option : ");
                scanf("%d",&op);
                switch(op) {
                        case 1:
                                printf("Enter element : ");
                                scanf("%d",&x);
                                enqueue(x);
                                break;
                        case 2:
                                dequeue();
                                break;
                        case 3:
                                display();
                                break;
                        case 4:
                                isEmpty();
                                break;
                        case 5:
                                size();
                                break;
                        case 6: exit(0);
                }
        }
}
```

CQueueOperations.c

ID: 22B01A4609
2022-2026-CSE-CS-A
Shri Vishnu Engineering College for Women (Autonomous)

```
#define max 5
int front = -1,rear =-1;
int que[max];
int val;
void enqueue(int val)
{
        if(front == 0 && rear == max-1)
        {
                printf("Circular queue is overflow.\n");
        }
        else if(front == -1 && rear == -1)
        {
                front = front + 1;
                rear = rear + 1;
                que[rear] = val;
                printf("Successfully inserted.\n");
        }
        else if(front != 0 && rear == max-1)
        {
                rear = 0;
                que[rear] = val;
                printf("Successfully inserted.\n");
        }
        else
        {
                rear = rear +1;
                que[rear] = val;
                printf("Successfully inserted.\n");

        }
}
void dequeue()
{
        int val;
        if(front == -1)
        {
                printf("Circular queue is underflow.\n");
        }
        else
        {
                if(rear == front)
                {
                        val = que[front];
                        rear = -1;
                        front = -1;
                        printf("Deleted element = %d\n",val);
                }
                else if(front == max-1)
                {
                        val = que[front];
                        front =front +1;
                        printf("Deleted element = %d\n",val);
                }
                else
                {
```

```
                        printf("Deleted element = %d\n",val);

                }
        }
}
void display()
{
        int i;
        if(front == -1)
        {
                printf("Circular queue is empty.\n");
        }
        else if(front > rear)
        {
                printf("Elements in the circular queue : ");
                for(i=front;i<max;i++)
                {
                        printf("%d ",que[i]);
                }
                for(i=0;i<=rear;i++)
                {
                        printf("%d ",que[i]);
                }
                printf("\n");
        }
        else
        {
                printf("Elements in the circular queue : ");
                for(i=front;i<=rear;i++)
                    printf("%d ",que[i]);
                printf("\n");
        }
}
void size()
{
        int count = 0,i;
        if(front == -1)
        {
                count = 0;
        }
        else if(front>rear)
        {
                for(i=front;i<max;i++)
                count = count +1;
        }
        else
        {
                for(i=front;i<=rear;i++)
                count = count +1;
        }
        printf("Circular queue size : %d\n",count);
}
void isEmpty()
{
        if(front == -1)
```

```
        }
        else
        {
                printf("Circular queue is not empty.\n");
        }
}
```

Execution Results - All test cases have succeeded!

Test Case - 1
User Output
1.Enqueue 2.Dequeue 3.Display 4.Is empty 5.Size 6.Exit
Enter your option :
1
Enter element :
34
Successfully inserted.
1.Enqueue 2.Dequeue 3.Display 4.Is empty 5.Size 6.Exit
Enter your option :
1
Enter element :
55
Successfully inserted.
1.Enqueue 2.Dequeue 3.Display 4.Is empty 5.Size 6.Exit
Enter your option :
1
Enter element :
26
Successfully inserted.
1.Enqueue 2.Dequeue 3.Display 4.Is empty 5.Size 6.Exit
Enter your option :
1
Enter element :
77
Successfully inserted.
1.Enqueue 2.Dequeue 3.Display 4.Is empty 5.Size 6.Exit
Enter your option :
1
Enter element :
38
Successfully inserted.
1.Enqueue 2.Dequeue 3.Display 4.Is empty 5.Size 6.Exit
Enter your option :
1
Enter element :
59
Circular queue is overflow.
1.Enqueue 2.Dequeue 3.Display 4.Is empty 5.Size 6.Exit

```
Enter your option :
3
Elements in the circular queue : 34 55 26 77 38
1.Enqueue 2.Dequeue 3.Display 4.Is empty 5.Size 6.Exit
Enter your option :
5
Circular queue size : 5
1.Enqueue 2.Dequeue 3.Display 4.Is empty 5.Size 6.Exit
Enter your option :
2
Deleted element = 34
1.Enqueue 2.Dequeue 3.Display 4.Is empty 5.Size 6.Exit
Enter your option :
2
Deleted element = 55
1.Enqueue 2.Dequeue 3.Display 4.Is empty 5.Size 6.Exit
Enter your option :
2
Deleted element = 26
1.Enqueue 2.Dequeue 3.Display 4.Is empty 5.Size 6.Exit
Enter your option :
3
Elements in the circular queue : 77 38
1.Enqueue 2.Dequeue 3.Display 4.Is empty 5.Size 6.Exit
Enter your option :
4
Circular queue is not empty.
1.Enqueue 2.Dequeue 3.Display 4.Is empty 5.Size 6.Exit
Enter your option :
5
Circular queue size : 2
1.Enqueue 2.Dequeue 3.Display 4.Is empty 5.Size 6.Exit
Enter your option :
6
```

Test Case - 2

User Output

```
1.Enqueue 2.Dequeue 3.Display 4.Is empty 5.Size 6.Exit
Enter your option :
2
Circular queue is underflow.
1.Enqueue 2.Dequeue 3.Display 4.Is empty 5.Size 6.Exit
Enter your option :
3
Circular queue is empty.
1.Enqueue 2.Dequeue 3.Display 4.Is empty 5.Size 6.Exit
Enter your option :
4
Circular queue is empty.
```

```
1.Enqueue 2.Dequeue 3.Display 4.Is empty 5.Size 6.Exit
Enter your option :
1
Enter element :
12
Successfully inserted.
1.Enqueue 2.Dequeue 3.Display 4.Is empty 5.Size 6.Exit
Enter your option :
1
Enter element :
34
Successfully inserted.
1.Enqueue 2.Dequeue 3.Display 4.Is empty 5.Size 6.Exit
Enter your option :
1
Enter element :
56
Successfully inserted.
1.Enqueue 2.Dequeue 3.Display 4.Is empty 5.Size 6.Exit
Enter your option :
3
Elements in the circular queue : 12 34 56
1.Enqueue 2.Dequeue 3.Display 4.Is empty 5.Size 6.Exit
Enter your option :
1
Enter element :
38
Successfully inserted.
1.Enqueue 2.Dequeue 3.Display 4.Is empty 5.Size 6.Exit
Enter your option :
1
Enter element :
25
Successfully inserted.
1.Enqueue 2.Dequeue 3.Display 4.Is empty 5.Size 6.Exit
Enter your option :
3
Elements in the circular queue : 12 34 56 38 25
1.Enqueue 2.Dequeue 3.Display 4.Is empty 5.Size 6.Exit
Enter your option :
1
Enter element :
56
Circular queue is overflow.
1.Enqueue 2.Dequeue 3.Display 4.Is empty 5.Size 6.Exit
Enter your option :
2
Deleted element = 12
1.Enqueue 2.Dequeue 3.Display 4.Is empty 5.Size 6.Exit
Enter your option :
```

2
Deleted element = 34
1.Enqueue 2.Dequeue 3.Display 4.Is empty 5.Size 6.Exit
Enter your option :
3
Elements in the circular queue : 56 38 25
1.Enqueue 2.Dequeue 3.Display 4.Is empty 5.Size 6.Exit
Enter your option :
4
Circular queue is not empty.
1.Enqueue 2.Dequeue 3.Display 4.Is empty 5.Size 6.Exit
Enter your option :
5
Circular queue size : 3
1.Enqueue 2.Dequeue 3.Display 4.Is empty 5.Size 6.Exit
Enter your option :
1
Enter element :
11
Successfully inserted.
1.Enqueue 2.Dequeue 3.Display 4.Is empty 5.Size 6.Exit
Enter your option :
3
Elements in the circular queue : 56 38 25 11
1.Enqueue 2.Dequeue 3.Display 4.Is empty 5.Size 6.Exit
Enter your option :
6

Test Case - 3
User Output
1.Enqueue 2.Dequeue 3.Display 4.Is empty 5.Size 6.Exit
Enter your option :
1
Enter element :
25
Successfully inserted.
1.Enqueue 2.Dequeue 3.Display 4.Is empty 5.Size 6.Exit
Enter your option :
1
Enter element :
63
Successfully inserted.
1.Enqueue 2.Dequeue 3.Display 4.Is empty 5.Size 6.Exit
Enter your option :
1
Enter element :
95
Successfully inserted.
1.Enqueue 2.Dequeue 3.Display 4.Is empty 5.Size 6.Exit

```
Enter your option :
1
Enter element :
42
Successfully inserted.
1.Enqueue 2.Dequeue 3.Display 4.Is empty 5.Size 6.Exit
Enter your option :
1
Enter element :
67
Successfully inserted.
1.Enqueue 2.Dequeue 3.Display 4.Is empty 5.Size 6.Exit
Enter your option :
1
Enter element :
54
Circular queue is overflow.
1.Enqueue 2.Dequeue 3.Display 4.Is empty 5.Size 6.Exit
Enter your option :
1
Enter element :
32
Circular queue is overflow.
1.Enqueue 2.Dequeue 3.Display 4.Is empty 5.Size 6.Exit
Enter your option :
2
Deleted element = 25
1.Enqueue 2.Dequeue 3.Display 4.Is empty 5.Size 6.Exit
Enter your option :
3
Elements in the circular queue : 63 95 42 67
1.Enqueue 2.Dequeue 3.Display 4.Is empty 5.Size 6.Exit
Enter your option :
4
Circular queue is not empty.
1.Enqueue 2.Dequeue 3.Display 4.Is empty 5.Size 6.Exit
Enter your option :
5
Circular queue size : 4
1.Enqueue 2.Dequeue 3.Display 4.Is empty 5.Size 6.Exit
Enter your option :
6
```

S.No: 10	Exp. Name: ***Singly linked list operations***	Date: 2023-06-15

Aim:

Write a menu driven **C** program to implement below listed operations on singly linked list.

1. Create 1- Way list
2. Insertion operation's (At front, At end, and At given position).
3. Deletion operation's (At front, At end, and At given position).
4. Search for an element
5. Reversing of 1- Way list created.

Note:

- For operation number 2 and 3 position entered by the user must be a valid position and first node must be considered at position 1.
- Partial code and visible test cases are given below for your reference.

Source Code:

SinglyLinkedList.c

```c
// C program for the all operations in the Singly Linked List

#include <stdio.h>
#include <stdlib.h>
// Linked List Node
struct node {
        int info;
        struct node*link;
}*start;
void createList() {
        if(start==NULL)
        {
                int n;
                printf("Enter the number of nodes: ");
                scanf("%d",&n);
                if(n!=0)
                {
                        int data;
                        struct node*newnode;
                        struct node*temp;
                        newnode=(struct node*)malloc(sizeof(struct node));
                        printf("Enter data to insert: ");
                        scanf("%d ",&data);
                        newnode->link==NULL;
                        start=newnode;
                        temp=start;
                        start->info=data;
                        for(int i=2;i<=n;i++)
                        {
                                newnode=(struct node*)malloc(sizeof(struct node));
                                scanf("%d",&data);
                                newnode->info=data;
                                newnode->link=NULL;
                                temp->link=newnode;
                                temp=temp->link;
                        }
                        printf("List is created\n");
                }
        }
        else
        printf("List is already created\n");
}
void traverse(){
        struct node *temp;
        if(start==NULL)
        printf("List is Empty\n");
        else{
                temp=start;
                while(temp!=NULL)
                {
                        printf("%d ",temp->info);
                        temp=temp->link;
                }
                printf("\n");
        }
```


ID: 22B01A4609

Shri Vishnu Engineering College for Women (Autonomous)

```
void insertAtFront() {
        int data;
        struct node *newnode;
        newnode=(struct node*)malloc(sizeof(struct node));
        printf("Enter data to insert: ");
        scanf("%d",&data);
        newnode->info=data;
        newnode->link=start;
        start=newnode;
}


void insertAtEnd() {
        int data;
        struct node*newnode,*head;
        newnode=(struct node*)malloc(sizeof(struct node));
        printf("Enter data to insert: ");
        scanf("%d",&data);
        newnode->link=NULL;
        newnode->info=data;
        head=start;
        while(head->link!=NULL)
        {
                head=head->link;
        }
        head->link=newnode;
}


void insertAtPosition() {
        struct node*temp,*newnode;
        int pos,data;
        newnode=(struct node*)malloc(sizeof(struct node));
        printf("Enter a valid position and data: ");
        scanf("%d %d",&pos,&data);
        newnode->info=data;
        temp=start;
        for(int i=1;i<pos-1;i++)
            temp=temp->link;
            newnode->link=temp->link;
            temp->link=newnode;
}

        //complete the code
void deleteFront(){
        struct node*temp;
        if(start!=NULL){
        temp=start;
        start=start->link;
        free(temp);
        }
}

void deleteEnd(){
        struct node*temp,*prevnode;
```

```
                temp=start;
            start=start->link;
            free(temp);
        }
        else{
                temp=start;
                while(temp->link!=NULL){
                        prevnode=temp;
                        temp=temp->link;
                }
                prevnode->link=NULL;
                free(temp);
        }
}


void deletePosition() {
        struct node*ptr,*ptr1;
        int i,pos;
        if(start!=NULL)
        {
                printf("Enter a valid position: ");
                scanf("%d",&pos);
                ptr=start;
                for(i=1;i<pos;i++)
                {
                        ptr1=ptr;
                        ptr=ptr->link;
                }
                ptr1->link=ptr->link;
                free(ptr);
        }
}

int searchPosOfEle(int key) {
        struct node*ptr=start;
        int i=0;
        if(ptr==NULL)
        {
                return i;
        }
        while(ptr!=NULL)
        {
                if(ptr->info==key)
        {
                return i+1;
        }
        i++;
        ptr=ptr->link;
        }
}

void reverseList() {
        struct node*prevnode,*curnode;
        prevnode=curnode=NULL;
```

```
        else{
                while(start!=NULL)
                {
                        curnode=start->link;
                        start->link=prevnode;
                        prevnode=start;
                        start=curnode;
                }
                start=prevnode;
                printf("reversed linked list: ");
                display();
        }
}
void display()
{
        struct node*temp;
        if(start==NULL)
        printf("List is Empty\n");
        else
        {
                temp=start;
                while(temp!=NULL)
                {
                        printf("%d ",temp->info);
                        temp=temp->link;
                }
                printf("\n");
        }
}
int main()
{
    int choice,x,pos;
    struct node *start = NULL;
    struct node *s1,*s2;
    while (1) {

                printf("1. create list\n");
        printf("2. insertion at front\n");
        printf("3. insertion at end\n");
        printf("4. insertion at given position\n");
        printf("5. deletion from front\n");
        printf("6. deletion from end\n");
        printf("7. deletion from given position\n");
        printf("8. Search an element\n");
        printf("9. reverse the list\n");
        printf("10. Exit\n");
        printf("Enter Choice: ");
        scanf("%d", &choice);

        switch (choice) {
        case 1:
            createList();
            traverse();
            break;
        case 2:
```

```
            break;
        case 3:
            insertAtEnd();
            traverse();
            break;
        case 4:
            insertAtPosition();
            traverse();
            break;
        case 5:
            deleteFront();
            traverse();
            break;
        case 6:
            deleteEnd();
            traverse();
            break;
        case 7:
            deletePosition();
            traverse();
            break;
        case 8:
            printf("Enter search element : ");
                    scanf("%d", &x);
                        pos = searchPosOfEle(x);
                        if (pos == 0) {
                                printf("The given element %d is not found in the given
SLL\n", x);
                        } else {
                                printf("The given element %d is found at position : %d\n",
x, pos);
                        }
                        break;
        case 9:
            reverseList();
            break;
        case 10:
            exit(1);
        default:
            printf("Invalid Choice\n");
        }
    }
    return 0;
}
```

Execution Results - All test cases have succeeded!

Test Case - 1
User Output
1. create list
2. insertion at front
3. insertion at end
4. insertion at given position

ID: 22B01A4609
2022-2026-CSE-CS-A
Shri Vishnu Engineering College for Women (Autonomous)

```
5. deletion from front
6. deletion from end
7. deletion from given position
8. Search an element
9. reverse the list
10. Exit
Enter Choice:
1
Enter the number of nodes:
5
Enter data to insert:
9 8 7 6 5
List is created
9 8 7 6 5
1. create list
2. insertion at front
3. insertion at end
4. insertion at given position
5. deletion from front
6. deletion from end
7. deletion from given position
8. Search an element
9. reverse the list
10. Exit
Enter Choice:
5
8 7 6 5
1. create list
2. insertion at front
3. insertion at end
4. insertion at given position
5. deletion from front
6. deletion from end
7. deletion from given position
8. Search an element
9. reverse the list
10. Exit
Enter Choice:
2
Enter data to insert:
9
9 8 7 6 5
1. create list
2. insertion at front
3. insertion at end
4. insertion at given position
5. deletion from front
6. deletion from end
7. deletion from given position
8. Search an element
9. reverse the list
10. Exit
```

```
Enter Choice:
6
9 8 7 6
1. create list
2. insertion at front
3. insertion at end
4. insertion at given position
5. deletion from front
6. deletion from end
7. deletion from given position
8. Search an element
9. reverse the list
10. Exit
Enter Choice:
3
Enter data to insert:
5
9 8 7 6 5
1. create list
2. insertion at front
3. insertion at end
4. insertion at given position
5. deletion from front
6. deletion from end
7. deletion from given position
8. Search an element
9. reverse the list
10. Exit
Enter Choice:
7
Enter a valid position:
3
9 8 6 5
1. create list
2. insertion at front
3. insertion at end
4. insertion at given position
5. deletion from front
6. deletion from end
7. deletion from given position
8. Search an element
9. reverse the list
10. Exit
Enter Choice:
4
Enter a valid position and data:
3 7
9 8 7 6 5
1. create list
2. insertion at front
3. insertion at end
4. insertion at given position
```

```
5. deletion from front
6. deletion from end
7. deletion from given position
8. Search an element
9. reverse the list
10. Exit
Enter Choice:
8
Enter search element :
6
The given element 6 is found at position : 4
1. create list
2. insertion at front
3. insertion at end
4. insertion at given position
5. deletion from front
6. deletion from end
7. deletion from given position
8. Search an element
9. reverse the list
10. Exit
Enter Choice:
8
Enter search element :
10
The given element 10 is not found in the given SLL
1. create list
2. insertion at front
3. insertion at end
4. insertion at given position
5. deletion from front
6. deletion from end
7. deletion from given position
8. Search an element
9. reverse the list
10. Exit
Enter Choice:
9
reversed linked list: 5 6 7 8 9
1. create list
2. insertion at front
3. insertion at end
4. insertion at given position
5. deletion from front
6. deletion from end
7. deletion from given position
8. Search an element
9. reverse the list
10. Exit
Enter Choice:
5
6 7 8 9
```

```
1. create list
2. insertion at front
3. insertion at end
4. insertion at given position
5. deletion from front
6. deletion from end
7. deletion from given position
8. Search an element
9. reverse the list
10. Exit
Enter Choice:
6
6 7 8
1. create list
2. insertion at front
3. insertion at end
4. insertion at given position
5. deletion from front
6. deletion from end
7. deletion from given position
8. Search an element
9. reverse the list
10. Exit
Enter Choice:
5
7 8
1. create list
2. insertion at front
3. insertion at end
4. insertion at given position
5. deletion from front
6. deletion from end
7. deletion from given position
8. Search an element
9. reverse the list
10. Exit
Enter Choice:
6
7
1. create list
2. insertion at front
3. insertion at end
4. insertion at given position
5. deletion from front
6. deletion from end
7. deletion from given position
8. Search an element
9. reverse the list
10. Exit
Enter Choice:
6
List is Empty
```

```
1. create list
2. insertion at front
3. insertion at end
4. insertion at given position
5. deletion from front
6. deletion from end
7. deletion from given position
8. Search an element
9. reverse the list
10. Exit
Enter Choice:
5
List is Empty
1. create list
2. insertion at front
3. insertion at end
4. insertion at given position
5. deletion from front
6. deletion from end
7. deletion from given position
8. Search an element
9. reverse the list
10. Exit
Enter Choice:
7
List is Empty
1. create list
2. insertion at front
3. insertion at end
4. insertion at given position
5. deletion from front
6. deletion from end
7. deletion from given position
8. Search an element
9. reverse the list
10. Exit
Enter Choice:
8
Enter search element :
0
The given element 0 is not found in the given SLL
1. create list
2. insertion at front
3. insertion at end
4. insertion at given position
5. deletion from front
6. deletion from end
7. deletion from given position
8. Search an element
9. reverse the list
10. Exit
Enter Choice:
```

9
List is empty
1. create list
2. insertion at front
3. insertion at end
4. insertion at given position
5. deletion from front
6. deletion from end
7. deletion from given position
8. Search an element
9. reverse the list
10. Exit
Enter Choice:
10

Test Case - 2
User Output
1. create list
2. insertion at front
3. insertion at end
4. insertion at given position
5. deletion from front
6. deletion from end
7. deletion from given position
8. Search an element
9. reverse the list
10. Exit
Enter Choice:
2
Enter data to insert:
10
10
1. create list
2. insertion at front
3. insertion at end
4. insertion at given position
5. deletion from front
6. deletion from end
7. deletion from given position
8. Search an element
9. reverse the list
10. Exit
Enter Choice:
1
List is already created
10
1. create list
2. insertion at front
3. insertion at end
4. insertion at given position

5. deletion from front
6. deletion from end
7. deletion from given position
8. Search an element
9. reverse the list
10. Exit
Enter Choice:
10

S.No: 11	Exp. Name: ***Write a C program to implement different Operations on Stack using Linked Lists***	**Date: 2023-05-31**

Aim:

Write a program to implement `stack` using **linked lists**.

```
Sample Input and Output:
    1.Push 2.Pop 3.Display 4.Is Empty 5.Peek 6.Exit
    Enter your option : 1
    Enter element : 33
    Successfully pushed.
    1.Push 2.Pop 3.Display 4.Is Empty 5.Peek 6.Exit
    Enter your option : 1
    Enter element : 22
    Successfully pushed.
    1.Push 2.Pop 3.Display 4.Is Empty 5.Peek 6.Exit
    Enter your option : 1
    Enter element : 55
    Successfully pushed.
    1.Push 2.Pop 3.Display 4.Is Empty 5.Peek 6.Exit
    Enter your option : 1
    Enter element : 66
    Successfully pushed.
    1.Push 2.Pop 3.Display 4.Is Empty 5.Peek 6.Exit
    Enter your option : 3
    Elements of the stack are : 66 55 22 33
    1.Push 2.Pop 3.Display 4.Is Empty 5.Peek 6.Exit
    Enter your option : 2
    Popped value = 66
    1.Push 2.Pop 3.Display 4.Is Empty 5.Peek 6.Exit
    Enter your option : 2
    Popped value = 55
    1.Push 2.Pop 3.Display 4.Is Empty 5.Peek 6.Exit
    Enter your option : 3
    Elements of the stack are : 22 33
    1.Push 2.Pop 3.Display 4.Is Empty 5.Peek 6.Exit
    Enter your option : 5
    Peek value = 22
    1.Push 2.Pop 3.Display 4.Is Empty 5.Peek 6.Exit
    Enter your option : 4
    Stack is not empty.
    1.Push 2.Pop 3.Display 4.Is Empty 5.Peek 6.Exit
    Enter your option : 6
```

Source Code:

StackUsingLL.c

ID: 22B01A4609
2022-2026-CSE-CS-A
Shri Vishnu Engineering College for Women (Autonomous)

```c
#include <stdio.h>
#include <stdlib.h>
#include "StackOperationsLL.c"

int main() {
        int op, x;
        while(1) {
                printf("1.Push 2.Pop 3.Display 4.Is Empty 5.Peek 6.Exit\n");
                printf("Enter your option : ");
                scanf("%d", &op);
                switch(op) {
                        case 1:
                                printf("Enter element : ");
                                scanf("%d", &x);
                                push(x);
                                break;
                        case 2:
                                pop();
                                break;
                        case 3:
                                display();
                                break;
                        case 4:
                                isEmpty();
                                break;
                        case 5:
                                peek();
                                break;
                        case 6:
                                exit(0);
                }
        }
}
```

StackOperationsLL.c

```
struct stack
{
        int data;
        struct stack * next;
};
typedef struct stack * stk;
stk top = NULL;
stk push(int ele)
{
        stk nn;
        nn = (stk)malloc(sizeof(stk));
        nn->data=ele;
        nn->next=top;
        top=nn;
        printf("Successfully pushed.\n");
        return top;
}
stk pop()
{
        stk temp;
        temp = top;
        if(temp==NULL)
        printf("Stack is underflow.\n");
        else
        {
                printf("Popped value = %d\n",temp->data);
                top=top->next;
                free(temp);
        }
}
void display()
{
        stk temp = top;
        if(temp==NULL)
        printf("Stack is empty.\n");
        else
        {
                printf("Elements of the stack are : ");
                while(temp!=NULL)
                {
                        printf("%d ",temp->data);
                        temp = temp->next;
                }
                printf("\n");
        }
}
void isEmpty()
{
        stk temp;
        temp = top;
        if(temp==NULL)
        {
                printf("Stack is empty.\n");
        }
        else
```

```
        }
}
stk peek()
{
        stk temp;
        temp=top;
        if(temp==NULL)
        {
                printf("Stack is underflow.\n");
        }
        else
        {
                printf("Peek value = %d\n",temp->data);
        }
}
```


Execution Results - All test cases have succeeded!

Test Case - 1
User Output
1.Push 2.Pop 3.Display 4.Is Empty 5.Peek 6.Exit
Enter your option :
1
Enter element :
33
Successfully pushed.
1.Push 2.Pop 3.Display 4.Is Empty 5.Peek 6.Exit
Enter your option :
1
Enter element :
22
Successfully pushed.
1.Push 2.Pop 3.Display 4.Is Empty 5.Peek 6.Exit
Enter your option :
1
Enter element :
55
Successfully pushed.
1.Push 2.Pop 3.Display 4.Is Empty 5.Peek 6.Exit
Enter your option :
1
Enter element :
66
Successfully pushed.
1.Push 2.Pop 3.Display 4.Is Empty 5.Peek 6.Exit
Enter your option :
3
Elements of the stack are : 66 55 22 33
1.Push 2.Pop 3.Display 4.Is Empty 5.Peek 6.Exit

```
Enter your option :
2
Popped value = 66
1.Push 2.Pop 3.Display 4.Is Empty 5.Peek 6.Exit
Enter your option :
2
Popped value = 55
1.Push 2.Pop 3.Display 4.Is Empty 5.Peek 6.Exit
Enter your option :
3
Elements of the stack are : 22 33
1.Push 2.Pop 3.Display 4.Is Empty 5.Peek 6.Exit
Enter your option :
5
Peek value = 22
1.Push 2.Pop 3.Display 4.Is Empty 5.Peek 6.Exit
Enter your option :
4
Stack is not empty.
1.Push 2.Pop 3.Display 4.Is Empty 5.Peek 6.Exit
Enter your option :
6
```

Test Case - 2

User Output

```
1.Push 2.Pop 3.Display 4.Is Empty 5.Peek 6.Exit
Enter your option :
2
Stack is underflow.
1.Push 2.Pop 3.Display 4.Is Empty 5.Peek 6.Exit
Enter your option :
3
Stack is empty.
1.Push 2.Pop 3.Display 4.Is Empty 5.Peek 6.Exit
Enter your option :
5
Stack is underflow.
1.Push 2.Pop 3.Display 4.Is Empty 5.Peek 6.Exit
Enter your option :
4
Stack is empty.
1.Push 2.Pop 3.Display 4.Is Empty 5.Peek 6.Exit
Enter your option :
1
Enter element :
23
Successfully pushed.
1.Push 2.Pop 3.Display 4.Is Empty 5.Peek 6.Exit
Enter your option :
```

```
1
Enter element :
24
Successfully pushed.
1.Push 2.Pop 3.Display 4.Is Empty 5.Peek 6.Exit
Enter your option :
3
Elements of the stack are : 24 23
1.Push 2.Pop 3.Display 4.Is Empty 5.Peek 6.Exit
Enter your option :
5
Peek value = 24
1.Push 2.Pop 3.Display 4.Is Empty 5.Peek 6.Exit
Enter your option :
2
Popped value = 24
1.Push 2.Pop 3.Display 4.Is Empty 5.Peek 6.Exit
Enter your option :
2
Popped value = 23
1.Push 2.Pop 3.Display 4.Is Empty 5.Peek 6.Exit
Enter your option :
2
Stack is underflow.
1.Push 2.Pop 3.Display 4.Is Empty 5.Peek 6.Exit
Enter your option :
4
Stack is empty.
1.Push 2.Pop 3.Display 4.Is Empty 5.Peek 6.Exit
Enter your option :
6
```

S.No: 12	Exp. Name: ***Write a C program to implement different Operations on Queue using Linked Lists***	**Date: 2023-06-08**

Aim:

Write a program to implement queue using **linked lists**.

```
Sample Input and Output:
    1.Enqueue 2.Dequeue 3.Display 4.Is Empty 5.Size 6.Exit
    Enter your option : 1
    Enter element : 57
    Successfully inserted.
    1.Enqueue 2.Dequeue 3.Display 4.Is Empty 5.Size 6.Exit
    Enter your option : 1
    Enter element : 87
    Successfully inserted.
    1.Enqueue 2.Dequeue 3.Display 4.Is Empty 5.Size 6.Exit
    Enter your option : 5
    Queue size : 2
    1.Enqueue 2.Dequeue 3.Display 4.Is Empty 5.Size 6.Exit
    Enter your option : 3
    Elements in the queue : 57 87
    1.Enqueue 2.Dequeue 3.Display 4.Is Empty 5.Size 6.Exit
    Enter your option : 2
    Deleted value = 57
    1.Enqueue 2.Dequeue 3.Display 4.Is Empty 5.Size 6.Exit
    Enter your option : 2
    Deleted value = 87
    1.Enqueue 2.Dequeue 3.Display 4.Is Empty 5.Size 6.Exit
    Enter your option : 3
    Queue is empty.
    1.Enqueue 2.Dequeue 3.Display 4.Is Empty 5.Size 6.Exit
    Enter your option : 5
    Queue size : 0
    1.Enqueue 2.Dequeue 3.Display 4.Is Empty 5.Size 6.Exit
    Enter your option : 6
```

Source Code:

QueueUsingLL.c

ID: 22B01A4609

Shri Vishnu Engineering College for Women (Autonomous)

```c
#include <conio.h>
#include <stdio.h>
#include "QueueOperationsLL.c"
int main() {
        int op, x;
        while(1) {
                printf("1.Enqueue 2.Dequeue 3.Display 4.Is Empty 5.Size 6.Exit\n");
                printf("Enter your option : ");
                scanf("%d",&op);
                switch(op) {
                        case 1:
                                printf("Enter element : ");
                                scanf("%d",&x);
                                enqueue(x);
                                break;
                        case 2:
                                dequeue();
                                break;
                        case 3:
                                display();
                                break;
                        case 4:
                                isEmpty();
                                break;
                        case 5:
                                size();
                                break;
                        case 6: exit(0);
                }
        }
}
```

QueueOperationsLL.c

```
struct queue
{
        int data;
        struct queue *next;
};
typedef struct queue *NODE;
NODE front=NULL,rear=NULL;
NODE enqueue(int ele)
{
        NODE nn;
        nn=(NODE)malloc(sizeof(NODE));
        nn->data=ele;
        nn->next=NULL;
        if(front==NULL)
        {
                front=nn;
                rear=nn;
        }
        else
        {
                rear->next=nn;
                rear=nn;
        }
        printf("Successfully inserted.\n");
}
NODE dequeue()
{
        NODE temp=front;
        if(front==NULL && rear==NULL)
        printf("Queue is underflow.\n");
        else
        {
                if(front==rear)
                {
                        front=NULL;
                        rear=NULL;
                }
                else
                {
                        front=front->next;
                }
                printf("Deleted value = %d\n",temp->data);
                free(temp);
        }
}
NODE display()
{
        NODE temp=front;
        if(front==NULL && rear==NULL)
        printf("Queue is empty.\n");
        else
        {
                printf("Elements in the queue : ");
                while(temp!=NULL)
                {
```

```
                }
                printf("\n");
        }
}
NODE isEmpty()
{
        if(front==NULL && rear ==NULL)
        printf("Queue is empty.\n");
        else
        printf("Queue is not empty.\n");
}
NODE size()
{
        NODE temp=front;
        int count=0;
        if(front==NULL && rear==NULL)
        printf("Queue size : 0\n");
        else
        {
                while(temp!=NULL)
                {
                        count++;
                        temp=temp->next;
                }
                printf("Queue size : %d\n",count);
        }
}
```

Execution Results - All test cases have succeeded!

Test Case - 1
User Output
1.Enqueue 2.Dequeue 3.Display 4.Is Empty 5.Size 6.Exit
Enter your option :
2
Queue is underflow.
1.Enqueue 2.Dequeue 3.Display 4.Is Empty 5.Size 6.Exit
Enter your option :
3
Queue is empty.
1.Enqueue 2.Dequeue 3.Display 4.Is Empty 5.Size 6.Exit
Enter your option :
4
Queue is empty.
1.Enqueue 2.Dequeue 3.Display 4.Is Empty 5.Size 6.Exit
Enter your option :
5
Queue size : 0
1.Enqueue 2.Dequeue 3.Display 4.Is Empty 5.Size 6.Exit
Enter your option :
1

Enter element :
44
Successfully inserted.
1.Enqueue 2.Dequeue 3.Display 4.Is Empty 5.Size 6.Exit
Enter your option :
1
Enter element :
55
Successfully inserted.
1.Enqueue 2.Dequeue 3.Display 4.Is Empty 5.Size 6.Exit
Enter your option :
1
Enter element :
66
Successfully inserted.
1.Enqueue 2.Dequeue 3.Display 4.Is Empty 5.Size 6.Exit
Enter your option :
1
Enter element :
67
Successfully inserted.
1.Enqueue 2.Dequeue 3.Display 4.Is Empty 5.Size 6.Exit
Enter your option :
3
Elements in the queue : 44 55 66 67
1.Enqueue 2.Dequeue 3.Display 4.Is Empty 5.Size 6.Exit
Enter your option :
2
Deleted value = 44
1.Enqueue 2.Dequeue 3.Display 4.Is Empty 5.Size 6.Exit
Enter your option :
2
Deleted value = 55
1.Enqueue 2.Dequeue 3.Display 4.Is Empty 5.Size 6.Exit
Enter your option :
5
Queue size : 2
1.Enqueue 2.Dequeue 3.Display 4.Is Empty 5.Size 6.Exit
Enter your option :
4
Queue is not empty.
1.Enqueue 2.Dequeue 3.Display 4.Is Empty 5.Size 6.Exit
Enter your option :
6

Test Case - 2
User Output
1.Enqueue 2.Dequeue 3.Display 4.Is Empty 5.Size 6.Exit
Enter your option :

1
Enter element :
23
Successfully inserted.
1.Enqueue 2.Dequeue 3.Display 4.Is Empty 5.Size 6.Exit
Enter your option :
1
Enter element :
234
Successfully inserted.
1.Enqueue 2.Dequeue 3.Display 4.Is Empty 5.Size 6.Exit
Enter your option :
1
Enter element :
45
Successfully inserted.
1.Enqueue 2.Dequeue 3.Display 4.Is Empty 5.Size 6.Exit
Enter your option :
1
Enter element :
456
Successfully inserted.
1.Enqueue 2.Dequeue 3.Display 4.Is Empty 5.Size 6.Exit
Enter your option :
2
Deleted value = 23
1.Enqueue 2.Dequeue 3.Display 4.Is Empty 5.Size 6.Exit
Enter your option :
3
Elements in the queue : 234 45 456
1.Enqueue 2.Dequeue 3.Display 4.Is Empty 5.Size 6.Exit
Enter your option :
2
Deleted value = 234
1.Enqueue 2.Dequeue 3.Display 4.Is Empty 5.Size 6.Exit
Enter your option :
3
Elements in the queue : 45 456
1.Enqueue 2.Dequeue 3.Display 4.Is Empty 5.Size 6.Exit
Enter your option :
4
Queue is not empty.
1.Enqueue 2.Dequeue 3.Display 4.Is Empty 5.Size 6.Exit
Enter your option :
5
Queue size : 2
1.Enqueue 2.Dequeue 3.Display 4.Is Empty 5.Size 6.Exit
Enter your option :
6

S.No: 13	Exp. Name: ***ALL operations in Circular linked list.***	Date: 2023-06-15

Aim:

Write a program that uses functions to perform the following operations on a Circular Linked List:

6. Insert At specified position
7. Traverse the List
8. Search
9. Delete
10. Exit

Source Code:

`allOperationsInCLL.c`

ID: 22B01A4609

2022-2026-CSE-CS-A

Shri Vishnu Engineering College for Women (Autonomous)

```
#include<stdio.h>
#include<stdlib.h>
struct node
{
        int data;
        struct node*next;
}*temp,*newnode,*head=NULL;
int insertpos()
{
        temp=head;
        int count=1,pos;
        printf("Enter a position : ");
        scanf("%d",&pos);
        newnode=(struct node*)malloc(sizeof(struct node));
        printf("Enter an element : ");
        scanf("%d",&newnode->data);
        newnode->next=newnode;
        if(head==NULL)
        {
                if(pos>1)
                printf("No such position in CLL so insertion is not possible\n");
                else
                head=newnode;
        return 0;
        }
        else if(pos==1)
        {
                newnode->next=head;
                temp=head;
                while(temp->next!=head)
                temp=temp->next;
                temp->next=newnode;head=newnode;
                return 0;
        }
        else
        {
                temp=head;
                while(count!=pos-1)
                {
                        temp=temp->next;
                        if(temp==head)
                        {
                                printf("No such position in CLL so insertion is not
possible\n");
                                return 0;
                        }
                        count++;
                }
                newnode->next=temp->next;
                temp->next=newnode;
                return 0;
        }
}
int deletepos()
{
```

```
        scanf("%d",&pos);
        if(head==NULL)
        {
                printf("No such position in CLL so deletion is not posssible\n");
        }
        else if(pos==1)
        {
                temp=head;
                printf("The deleted element from CLL : %d\n",head->data);
                while(temp->next!=NULL)
                     temp=temp->next;
                     head=head->next;
                     if(head==temp)
                     head=NULL;
                     else
                     temp->next=head;
                     return 0;
        }
        else
        {
                temp=head;
                while(count!=pos-1)
                {
                        temp=temp->next;
                        if(temp->next==head)
                        {
                                printf("No such position in CLL so deletion is not
possible\n");
                                return 0;
                        }
                        count++;
                }
                if(temp->next==head)
                {
                        printf("No such position in CLL so deletion is not possible\n");
                        return 0;
                }
                printf("The deleted element from CLL : %d\n",temp->next->data);
                struct node*delnode=temp->next;
                temp->next=delnode->next;
                return 0;
        }
}
void display()
{
        if(head==NULL)
        printf("The circular linked list os empty\n");
        else
        {
                printf("The elements in CLL are : ");
                temp=head;
                while(temp->next!=head)
                {
                        printf("%d --> ",temp->data);
                        temp=temp->next;
```

```
        }
}
int search(int x)
{
        int count=0,ele=0;
        temp=head;
        if(head==NULL)
        return 0;
        else if(x==temp->data)
        return 1;
        else
        {
                while(temp->next!=head)
                {
                        if(x==temp->data)
                        {
                                count=count+1;
                                return count;
                        }
                        else
                        {
                                temp=temp->next;
                                count++;
                        }
                }
                if(x==temp->data)
                return count+1;
                return ele;
        }
}
int main()
{
        int opt,x,pos;
        while(1)
        {
                printf("1. Insert At specified position\n2. Traverse the List\n3. Search\n4.
Delete\n5. Exit\nEnter your option : ");
                scanf("%d",&opt);
                switch(opt)
                {
                        case 1: insertpos();
                                break;
                        case 2: display();
                                break;
                        case 3:printf("Enter search element : ");
                               scanf("%d",&x);
                               pos=search(x);
                               if(pos==0)
                               printf("The given element %d is not found in the given
CLL\n",x);
                               else
                               printf("The given element %d is found at position :
%d\n",x,pos);
                               break;
                    case 4:deletepos();
```

```
                }
        }
}
```

Execution Results - All test cases have succeeded!

Test Case - 1
User Output
1. Insert At specified position
2. Traverse the List
3. Search
4. Delete
5. Exit
Enter your option :
1
Enter a position :
1
Enter an element :
12
1. Insert At specified position
2. Traverse the List
3. Search
4. Delete
5. Exit
Enter your option :
1
Enter a position :
1
Enter an element :
13
1. Insert At specified position
2. Traverse the List
3. Search
4. Delete
5. Exit
Enter your option :
1
Enter a position :
2
Enter an element :
14
1. Insert At specified position
2. Traverse the List
3. Search
4. Delete
5. Exit
Enter your option :
1

```
Enter a position :
3
Enter an element :
15
1. Insert At specified position
2. Traverse the List
3. Search
4. Delete
5. Exit
Enter your option :
2
The elements in CLL are : 13 --> 14 --> 15 --> 12 -->
1. Insert At specified position
2. Traverse the List
3. Search
4. Delete
5. Exit
Enter your option :
4
Enter position :
2
The deleted element from CLL : 14
1. Insert At specified position
2. Traverse the List
3. Search
4. Delete
5. Exit
Enter your option :
3
Enter search element :
12
The given element 12 is found at position : 3
1. Insert At specified position
2. Traverse the List
3. Search
4. Delete
5. Exit
Enter your option :
2
The elements in CLL are : 13 --> 15 --> 12 -->
1. Insert At specified position
2. Traverse the List
3. Search
4. Delete
5. Exit
Enter your option :
5
```

Test Case - 2

User Output

```
1. Insert At specified position
2. Traverse the List
3. Search
4. Delete
5. Exit
Enter your option :
1
Enter a position :
1
Enter an element :
1
1. Insert At specified position
2. Traverse the List
3. Search
4. Delete
5. Exit
Enter your option :
1
Enter a position :
2
Enter an element :
2
1. Insert At specified position
2. Traverse the List
3. Search
4. Delete
5. Exit
Enter your option :
1
Enter a position :
3
Enter an element :
3
1. Insert At specified position
2. Traverse the List
3. Search
4. Delete
5. Exit
Enter your option :
1
Enter a position :
4
Enter an element :
4
1. Insert At specified position
2. Traverse the List
3. Search
4. Delete
5. Exit
Enter your option :
1
```

```
Enter a position :
5
Enter an element :
5
1. Insert At specified position
2. Traverse the List
3. Search
4. Delete
5. Exit
Enter your option :
1
Enter a position :
6
Enter an element :
6
1. Insert At specified position
2. Traverse the List
3. Search
4. Delete
5. Exit
Enter your option :
2
The elements in CLL are : 1 --> 2 --> 3 --> 4 --> 5 --> 6 -->
1. Insert At specified position
2. Traverse the List
3. Search
4. Delete
5. Exit
Enter your option :
5
```

S.No: 14	Exp. Name: ***C program which performs all operations on double linked list.***	Date: 2023-06-12

Aim:

Write a **C** program that uses functions to perform the following **operations on double linked list.**
i) Creationii) Insertioniii) Deletioniv) Search v) Traversal

Source Code:

AllOperationsDLL.c

```
#include<stdio.h>
struct node
{
        int data;
        struct node*prev,*next;
}*head=NULL,*newnode,*temp;
void insertbeg()
{
        newnode=(struct node*)malloc(sizeof(struct node));
        printf("Enter an element: ");
        scanf("%d",&newnode->data);
        newnode->prev=NULL;
        newnode->next=NULL;
        if(head==NULL)
        head=newnode;
        else
        {
                head->prev=newnode;
                newnode->next=head;
                head=newnode;
        }
}
void deletebeg()
{
        if(head==NULL)
        printf("Double Linked List is empty so deletion is not possible\n");
        else
        {
                printf("The deleted element from DLL : %d\n",head->data);
                head=head->next;
        }
}
int search(int search)
{
        int ele=0,count=0;
        temp=head;
        if(head==NULL)
        return 0;
        else if(search==temp->data)
        return 1;
        else
        {
                while(temp->next!=NULL)
                {
                        if(search==temp->data)
                        {
                                count=count+1;
                                return count;
                        }
                        else
                        {
                                temp=temp->next;
                                count++;
                        }
                }
```

```
                return ele;
        }
}
void traverse()
{
        if(head==NULL)
        printf("Double Linked List is empty\n");
        else
        {
                printf("The elements in DLL are: ");
                temp=head;
                while(temp!=NULL)
                {
                        printf("%d <--> ",temp->data);
                        temp=temp->next;
                }
                printf("NULL\n");
        }
}
int main()
{
        int ch,sea,pos;
        while(1)
        {
                printf("1.Insert At Begin\n2.Delete at Begin\n3.Search an element
Position\n4.Traverse the List\n5.Exit\n");
                printf("Enter your option : ");
                scanf("%d",&ch);
                switch(ch)
                {
                        case 1:insertbeg();
                                break;
                        case 2:deletebeg();
                                break;
                        case 3:printf("Enter search element: ");
                               scanf("%d",&sea);
                               pos=search(sea);
                               if(pos==0)
                               printf("The given element %d is not found in the given
DLL\n",sea);
                               else
                               printf("The given element %d is found at position :
%d\n",sea,pos);
                               break;
                        case 4:traverse();
                               break;
                        case 5:exit(1);
                }
        }
}
```

Execution Results - All test cases have succeeded!

Test Case - 1

User Output
1.Insert At Begin
2.Delete at Begin
3.Search an element Position
4.Traverse the List
5.Exit
Enter your option :
1
Enter an element:
15
1.Insert At Begin
2.Delete at Begin
3.Search an element Position
4.Traverse the List
5.Exit
Enter your option :
2
The deleted element from DLL : 15
1.Insert At Begin
2.Delete at Begin
3.Search an element Position
4.Traverse the List
5.Exit
Enter your option :
1
Enter an element:
12
1.Insert At Begin
2.Delete at Begin
3.Search an element Position
4.Traverse the List
5.Exit
Enter your option :
1
Enter an element:
16
1.Insert At Begin
2.Delete at Begin
3.Search an element Position
4.Traverse the List
5.Exit
Enter your option :
1
Enter an element:
17
1.Insert At Begin
2.Delete at Begin
3.Search an element Position
4.Traverse the List
5.Exit
Enter your option :

4
The elements in DLL are: 17 <--> 16 <--> 12 <--> NULL
1.Insert At Begin
2.Delete at Begin
3.Search an element Position
4.Traverse the List
5.Exit
Enter your option :
3
Enter search element:
16
The given element 16 is found at position : 2
1.Insert At Begin
2.Delete at Begin
3.Search an element Position
4.Traverse the List
5.Exit
Enter your option :
4
The elements in DLL are: 17 <--> 16 <--> 12 <--> NULL
1.Insert At Begin
2.Delete at Begin
3.Search an element Position
4.Traverse the List
5.Exit
Enter your option :
5

Test Case - 2
User Output
1.Insert At Begin
2.Delete at Begin
3.Search an element Position
4.Traverse the List
5.Exit
Enter your option :
2
Double Linked List is empty so deletion is not possible
1.Insert At Begin
2.Delete at Begin
3.Search an element Position
4.Traverse the List
5.Exit
Enter your option :
4
Double Linked List is empty
1.Insert At Begin
2.Delete at Begin
3.Search an element Position
4.Traverse the List

```
5.Exit
Enter your option :
1
Enter an element:
101
1.Insert At Begin
2.Delete at Begin
3.Search an element Position
4.Traverse the List
5.Exit
Enter your option :
1
Enter an element:
102
1.Insert At Begin
2.Delete at Begin
3.Search an element Position
4.Traverse the List
5.Exit
Enter your option :
1
Enter an element:
103
1.Insert At Begin
2.Delete at Begin
3.Search an element Position
4.Traverse the List
5.Exit
Enter your option :
6
1.Insert At Begin
2.Delete at Begin
3.Search an element Position
4.Traverse the List
5.Exit
Enter your option :
2
The deleted element from DLL : 103
1.Insert At Begin
2.Delete at Begin
3.Search an element Position
4.Traverse the List
5.Exit
Enter your option :
4
The elements in DLL are: 102 <--> 101 <--> NULL
1.Insert At Begin
2.Delete at Begin
3.Search an element Position
4.Traverse the List
5.Exit
```

Enter your option :
3
Enter search element:
103
The given element 103 is not found in the given DLL
1.Insert At Begin
2.Delete at Begin
3.Search an element Position
4.Traverse the List
5.Exit
Enter your option :
4
The elements in DLL are: 102 <--> 101 <--> NULL
1.Insert At Begin
2.Delete at Begin
3.Search an element Position
4.Traverse the List
5.Exit
Enter your option :
5

S.No: 15	Exp. Name: ***Program to insert into BST and traversal using In-order, Pre-order and Post-order***	**Date: 2023-06-08**

Aim:

Write a program to create a binary search tree of integers and perform the following operations using linked list.

11. Insert a node
12. In-order traversal
13. Pre-order traversal
14. Post-order traversal

Source Code:

BinarySearchTree.c

ID: 22B01A4609

Shri Vishnu Engineering College for Women (Autonomous)

```
#include<stdio.h>
#include<stdlib.h>
#include "InsertAndTraversals.c"

void main() {
        int x, op;
        BSTNODE root = NULL;
        while(1) {
                printf("1.Insert 2.Inorder Traversal 3.Preorder Traversal 4.Postorder
Traversal 5.Exit\n");
                printf("Enter your option : ");
                scanf("%d", &op);
                switch(op) {
                        case 1: printf("Enter an element to be inserted : ");
                                        scanf("%d", &x);
                                        root = insertNodeInBST(root,x);
                                        break;
                        case 2:
                                        if(root == NULL) {
                                                printf("Binary Search Tree is empty.\n");
                                        }
                                        else {
                                                printf("Elements of the BST (in-order
traversal): ");
                                                inorderInBST(root);
                                                printf("\n");
                                                }
                                        break;
                        case 3:
                                        if(root == NULL) {
                                                printf("Binary Search Tree is empty.\n");
                                        }
                                        else {
                                                printf("Elements of the BST (pre-order
traversal): ");
                                                preorderInBST(root);
                                                printf("\n");
                                                }
                                        break;
                        case 4:
                                        if(root == NULL) {
                                                printf("Binary Search Tree is empty.\n");
                                        }
                                        else {
                                                printf("Elements of the BST (post-order
traversal): ");
                                                postorderInBST(root);
                                                printf("\n");
                                                }
                                        break;
                        case 5:
                                        exit(0);
                }
        }
}
```


ID: 22B01A4609

Shri Vishnu Engineering College for Women (Autonomous)

InsertAndTraversals.c

```
struct node {
        int data;
        struct node *left, *right;
};

typedef struct node *BSTNODE;

BSTNODE newNodeInBST(int item) {
        BSTNODE temp =  (BSTNODE)malloc(sizeof(struct node));
        temp->data = item;
        temp->left = temp->right = NULL;
        return temp;
}

void inorderInBST(BSTNODE root) {
        BSTNODE node=root;
        if(node==NULL)
        printf("Binary search tree is empty.\n");
        else
        {
                if(node->left!=NULL)
                inorderInBST(node->left);
                printf("%d ",node->data);
                if(node->right!=NULL)
                inorderInBST(node->right);
        }
}

void preorderInBST(BSTNODE root) {
        BSTNODE node=root;
        if(node==NULL)
        printf("Binary Search tree is empty.\n");
        else
        {
                printf("%d ",node->data);
                if(node->left!=NULL)
                preorderInBST(node->left);
                if(node->right!=NULL)
                preorderInBST(node->right);
        }
}
void postorderInBST(BSTNODE root) {
        BSTNODE node=root;
        if(root == NULL)
        printf("Binary Search is empty.\n");
        else
        {
                if(node->left!=NULL)
                postorderInBST(node->left);
                if(node->right!=NULL)
                postorderInBST(node->right);
                printf("%d ",node->data);
        }
}
BSTNODE insertNodeInBST(BSTNODE node, int ele) {
```

```
        if(node==NULL)
        {
                printf("Successfully inserted.\n");
                return nn;
        }
        else
        {
                if(ele<node->data)
                node->left=insertNodeInBST(node->left,ele);
                else if (ele>node->data)
                node->right=insertNodeInBST(node->right,ele);
                else
                printf("Element already exists in BST.\n");
        }
}
```

Execution Results - All test cases have succeeded!

Test Case - 1
User Output
1.Insert 2.Inorder Traversal 3.Preorder Traversal 4.Postorder Traversal 5.Exit
Enter your option :
1
Enter an element to be inserted :
54
Successfully inserted.
1.Insert 2.Inorder Traversal 3.Preorder Traversal 4.Postorder Traversal 5.Exit
Enter your option :
1
Enter an element to be inserted :
28
Successfully inserted.
1.Insert 2.Inorder Traversal 3.Preorder Traversal 4.Postorder Traversal 5.Exit
Enter your option :
1
Enter an element to be inserted :
62
Successfully inserted.
1.Insert 2.Inorder Traversal 3.Preorder Traversal 4.Postorder Traversal 5.Exit
Enter your option :
2
Elements of the BST (in-order traversal): 28 54 62
1.Insert 2.Inorder Traversal 3.Preorder Traversal 4.Postorder Traversal 5.Exit
Enter your option :
3
Elements of the BST (pre-order traversal): 54 28 62
1.Insert 2.Inorder Traversal 3.Preorder Traversal 4.Postorder Traversal 5.Exit
Enter your option :
4

```
Elements of the BST (post-order traversal): 28 62 54
1.Insert 2.Inorder Traversal 3.Preorder Traversal 4.Postorder Traversal 5.Exit
Enter your option :
5
```

Test Case - 2

User Output

```
1.Insert 2.Inorder Traversal 3.Preorder Traversal 4.Postorder Traversal 5.Exit
Enter your option :
1
Enter an element to be inserted :
100
Successfully inserted.
1.Insert 2.Inorder Traversal 3.Preorder Traversal 4.Postorder Traversal 5.Exit
Enter your option :
1
Enter an element to be inserted :
20
Successfully inserted.
1.Insert 2.Inorder Traversal 3.Preorder Traversal 4.Postorder Traversal 5.Exit
Enter your option :
1
Enter an element to be inserted :
200
Successfully inserted.
1.Insert 2.Inorder Traversal 3.Preorder Traversal 4.Postorder Traversal 5.Exit
Enter your option :
1
Enter an element to be inserted :
10
Successfully inserted.
1.Insert 2.Inorder Traversal 3.Preorder Traversal 4.Postorder Traversal 5.Exit
Enter your option :
1
Enter an element to be inserted :
30
Successfully inserted.
1.Insert 2.Inorder Traversal 3.Preorder Traversal 4.Postorder Traversal 5.Exit
Enter your option :
1
Enter an element to be inserted :
150
Successfully inserted.
1.Insert 2.Inorder Traversal 3.Preorder Traversal 4.Postorder Traversal 5.Exit
Enter your option :
1
Enter an element to be inserted :
300
Successfully inserted.
```

1.Insert 2.Inorder Traversal 3.Preorder Traversal 4.Postorder Traversal 5.Exit
Enter your option :
2
Elements of the BST (in-order traversal): 10 20 30 100 150 200 300
1.Insert 2.Inorder Traversal 3.Preorder Traversal 4.Postorder Traversal 5.Exit
Enter your option :
3
Elements of the BST (pre-order traversal): 100 20 10 30 200 150 300
1.Insert 2.Inorder Traversal 3.Preorder Traversal 4.Postorder Traversal 5.Exit
Enter your option :
4
Elements of the BST (post-order traversal): 10 30 20 150 300 200 100
1.Insert 2.Inorder Traversal 3.Preorder Traversal 4.Postorder Traversal 5.Exit
Enter your option :
5

S.No: 16	Exp. Name: ***Breadth First Search (BFS)***	Date: 2023-06-12

Aim:

Write a program to implement Breadth First Search (BFS) graph traversal methods.

Source Code:

GraphsBFS.c

ID: 22B01A4609

```
#include <stdio.h>
#include <stdlib.h>
#define MAX 99
struct node {
        struct node *next;
        int vertex;
};
typedef struct node * GNODE;
GNODE graph[20];
int visited[20];
int queue[MAX], front = -1,rear = -1;
int n;

void insertQueue(int vertex) {
        if(rear==MAX-1)
        {
                printf("Queue is full.\n");
        }
        else{
                if(front==-1)
                front=0;
                rear++;
                queue[rear]=vertex;
        }

}

int isEmptyQueue() {
        if(front==-1 || front>rear)
        return 1;
        else{
                return 0;
        }

}

int deleteQueue() {
        int ele;
        if(front == -1 || front>rear)
        {
                printf("Queue is empty.\n");
                return -1;
        }
        else{
                ele=queue[front];
                front++;
                return ele;
        }
}

void BFS(int v) {
        GNODE p;
        insertQueue(v);
        visited[v]=1;
        printf("%d\n",v);
```

```
                v=deleteQueue(v);
                for(p=graph[v];p!=NULL;p=p->next)
                {
                        if(visited[p->vertex]==0)
                        {
                                insertQueue(p->vertex);
                                visited[p->vertex]=1;
                                printf("%d\n",p->vertex);
                }
        }
  }
}

void main() {
        int N, E, s, d, i, j, v;
        GNODE p, q;
        printf("Enter the number of vertices : ");
        scanf("%d",&N);
        printf("Enter the number of edges : ");
        scanf("%d",&E);
        for(i=1;i<=E;i++) {
                printf("Enter source : ");
                scanf("%d",&s);
                printf("Enter destination : ");
                scanf("%d",&d);
                q=(GNODE)malloc(sizeof(struct node));
                q->vertex=d;
                q->next=NULL;
                if(graph[s]==NULL) {
                        graph[s]=q;
                } else {
                        p=graph[s];
                        while(p->next!=NULL)
                                p=p->next;
                        p->next=q;
                }
        }
        for(i=1;i<=n;i++)
                visited[i]=0;
        printf("Enter Start Vertex for BFS : ");
        scanf("%d", &v);
        printf("BFS of graph : \n");
        BFS(v);
        printf("\n");
}
```

Execution Results - All test cases have succeeded!

Test Case - 1
User Output
Enter the number of vertices :
5
Enter the number of edges :

5
Enter source :
1
Enter destination :
2
Enter source :
1
Enter destination :
4
Enter source :
4
Enter destination :
2
Enter source :
2
Enter destination :
3
Enter source :
4
Enter destination :
5
Enter Start Vertex for BFS :
1
BFS of graph :
1
2
4
3
5

Test Case - 2
User Output
Enter the number of vertices :
4
Enter the number of edges :
3
Enter source :
1
Enter destination :
2
Enter source :
2
Enter destination :
3
Enter source :
3
Enter destination :
4

```
Enter Start Vertex for BFS :
2
BFS of graph :
2
3
4
```

Test Case - 3

User Output

```
Enter the number of vertices :
9
Enter the number of edges :
12
Enter source :
0
Enter destination :
1
Enter source :
0
Enter destination :
3
Enter source :
0
Enter destination :
4
Enter source :
1
Enter destination :
2
Enter source :
1
Enter destination :
3
Enter source :
3
Enter destination :
6
Enter source :
6
Enter destination :
4
Enter source :
6
Enter destination :
7
Enter source :
7
Enter destination :
```

```
8
Enter source :
4
Enter destination :
5
Enter source :
2
Enter destination :
5
Enter source :
7
Enter destination :
5
Enter Start Vertex for BFS :
0
BFS of graph :
0
1
3
4
2
6
5
7
8
```

S.No: 17	Exp. Name: ***Depth First Search (DFS)***	Date: 2023-06-12

Aim:

Write a program to implement Depth First Search (DFS) graph traversal methods.

Source Code:

GraphsDFS.c

```
#include<stdio.h>
#include<stdlib.h>
struct node {
    struct node *next;
    int vertex;
};
typedef struct node * GNODE;
GNODE graph[20];
int visited[20];
int n;

void DFS(int i) {
        GNODE p;
        printf("%d\n",i);
        p=graph[i];
        visited[i]=1;
        while(p!=NULL)
        {
                i=p->vertex;
                if(visited[i]!=1)
                DFS(i);
                p=p->next;
        }

}

void main() {
    int N,E,i,s,d,sv;
    GNODE q,p;
    printf("Enter the number of vertices : ");
        scanf("%d",&N);
        printf("Enter the number of edges : ");
        scanf("%d",&E);
        for(i=1;i<=E;i++) {
                printf("Enter source : ");
                scanf("%d",&s);
                printf("Enter destination : ");
                scanf("%d",&d);
                q=(GNODE)malloc(sizeof(struct node));
        q->vertex=d;
        q->next=NULL;
        if(graph[s]==NULL)
                graph[s]=q;
        else {
                p=graph[s];
                while(p->next!=NULL)
                p=p->next;
                p->next=q;
        }
        }
        printf("Enter Start Vertex for DFS : ");
        scanf("%d",&sv);
    for(i=0;i<n;i++)
        visited[i]=0;
    printf("DFS of graph : \n");
```

```
    DFS(sv);
}
```

Execution Results - All test cases have succeeded!

Test Case - 1
User Output
Enter the number of vertices :
6
Enter the number of edges :
7
Enter source :
1
Enter destination :
2
Enter source :
1
Enter destination :
4
Enter source :
4
Enter destination :
2
Enter source :
2
Enter destination :
3
Enter source :
4
Enter destination :
5
Enter source :
4
Enter destination :
3
Enter source :
3
Enter destination :
6
Enter Start Vertex for DFS :
1
DFS of graph :
1
2
3
6
4

ID: 22B01A4609

Shri Vishnu Engineering College for Women (Autonomous)

5

Test Case - 2
User Output
Enter the number of vertices :
5
Enter the number of edges :
5
Enter source :
1
Enter destination :
2
Enter source :
1
Enter destination :
4
Enter source :
4
Enter destination :
2
Enter source :
2
Enter destination :
3
Enter source :
4
Enter destination :
5
Enter Start Vertex for DFS :
1
DFS of graph :
1
2
3
4
5

Test Case - 3
User Output
Enter the number of vertices :
4
Enter the number of edges :
4
Enter source :
1
Enter destination :
1

Enter source :
1
Enter destination :
3
Enter source :
2
Enter destination :
2
Enter source :
4
Enter destination :
3
Enter Start Vertex for DFS :
1
DFS of graph :
1
3