Name: MOHD ADIL

Roll No: 20BCS042

Branch: Computer Engineering

Subject: Data Structure Lab

Subject code: CEN 391

Subject Teachers:

Dr. Shahzad Alam **Mr. Faiyaz Ahmad**

S.No	Program Name	Date
1.	Write a menu driven program having the following option. Make user defined functions for each option. 1. Factorial of a Given no 2. Sum of Natural Series up to n terms 3. Print Fibonacci Series up to n terms 4. Power of a and b 5 exit	14th Sept
2.	1. Write a program to sort n elements of array using bubble sort.Show Each Iteration of bubble sort. 2. Write a program to sort n elements of array using Early termination bubble sort.Show Each Iteration of bubble sort.	21st Sept
3.	Write a menu program to maintain record of 10 employees in structure.	5th Oct
4.	Write a menu program to maintain record of employees in structure.(Take n from user and allocate memory to structure pointer dynamically) . Before Giving the menu to the user first ask the value of n.	12th Oct
5.	Write a menu program to maintain records of employees in structure.(allocate memory to structure pointer dynamically when needed by the user).	26th Oct
6.	1. Write a menu driven program to implement stack operations using Array. Initially take size of array(Take 5) from the user at run time and allocate the memory to array. 2. Write a menu driven program to implement stack operation using linked lists.	2nd Nov
7.	Write a menu driven program to implement normal Queue operations using Array.	9th Nov
8.	Write a menu driven program to implement Circular Queue operations using Array.	16th Nov
9.	Write a menu driven program to implement Simple Queue operations using Linked List.	23rd Nov
10	Write a menu driven program to implement Priority Queue operations using Linked List.	30th Nov
11.	Write a menu driven program to implement Singly Linked List having following operations.	7th Dec
12.	Write a menu driven program to implement doubly Linked List having following operations.	14th Dec

Program 1

```c
#include <stdio.h>
#include <stdlib.h>
int factorial(int n)
{
    int f = 1;
    for (int i = 1; i <= n; i++)
    {
        f = f * i;
    }
    return f;
}
int SumN(int n)
{
    int sum = 0;
    printf("The series is :");
    for (int i = 0; i < n; i++)
    {
        printf("%d ", i + 1);
        sum += i + 1;
    }
    printf("\n");
    return sum;
}
void fibonacci()
{
    int sum = 0, n;
    int a = 0;
    int b = 1;
    printf("Enter the value upto which Fibonacci series is to be printed:
");
    scanf("%d", &n);
    printf("Fibonacci series: ");

    while (sum <= n)
    {
        printf("%d ", sum);
        a = b;
        b = sum;
        sum = a + b;
    }
}
int power(int a, int b)
{
    int pow = 1;
```

```
    for (int i = 1; i <= b; i++)
    {
        pow = pow * a;
    }
    return pow;
}

int main()
{
    while (1)
    {
        int ch;
        printf("Enter 1 for Factorial\n");
        printf("Enter 2 for Sum of Series of natural number upto n
terms\n");
        printf("Enter 3 for Fibonacci\n");
        printf("Enter 4 for Power of a and b\n");
        printf("Enter your Choice: ");
        scanf("%d", &ch);
        switch (ch)
        {

        case 1:
            printf("\n*******************************\n");
            printf("You are in case 1\n");
            printf("Enter the number whose factorial is to be calculated:
");
            int n;
            scanf("%d", &n);
            printf("The factorial of %d is %d\n", n, factorial(n));
            printf("\n*******************************\n");
            break;
        case 2:
            printf("\n*******************************\n");
            printf("You are in case 2\n");
            printf("Enter the nth term of Natural Series: ");
            int m;
            scanf("%d", &m);
            printf("The sum of series is %d\n", SumN(m));
            printf("\n*******************************\n");
            break;
        case 3:
            printf("\n*******************************\n");
            printf("You are in case 3\n");
            fibonacci();
            printf("\n");
```

```c
            printf("\n*******************************\n");
            break;
        case 4:
            printf("\n*******************************\n");
            printf("You are in case 4\n");
            int a, b;
            printf("Enter the value of a and b: ");
            scanf("%d %d", &a, &b);

            printf("The value of %d to the power %d is : %d\n", a, b,
power(a, b));
            printf("\n*******************************\n");
            break;
        case 5:
            printf("\n*******************************\n");
            printf("Exiting...\n");
            exit(0);
        default:
            printf("\n*******************************\n");
            printf("Invalid Input\n");
            printf("\n*******************************\n");
        }
    }
    return 0;
}
```

Output:

```
PS C:\Users\aadil\Desktop\lab> cd "c:\Users\aadil\Desktop\lab\" ; if ($?) { gcc dsalab.c -o dsalab } ; if ($?)
Enter 1 for Factorial
Enter 2 for Sum of Series of natural number upto n terms
Enter 3 for Fibonacci
Enter 4 for Power of a and b
Enter your Choice: 1

*******************************
You are in case 1
Enter the number whose factorial is to be calculated: 5
The factorial of 5 is 120

*******************************
Enter 1 for Factorial
Enter 2 for Sum of Series of natural number upto n terms
Enter 3 for Fibonacci
Enter 4 for Power of a and b
Enter your Choice: 2

*******************************
You are in case 2
Enter the nth term of Natural Series: 25
The series is :1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25
The sum of series is 325

*******************************
Enter 1 for Factorial
Enter 2 for Sum of Series of natural number upto n terms
Enter 3 for Fibonacci
Enter 4 for Power of a and b
Enter your Choice: 3

*******************************
You are in case 3
Enter the value upto which Fibonacci series is to be printed: 1000
Fibonacci series: 0 1 1 2 3 5 8 13 21 34 55 89 144 233 377 610 987
```

```
*******************************
Enter 1 for Factorial
Enter 2 for Sum of Series of natural number upto n terms
Enter 3 for Fibonacci
Enter 4 for Power of a and b
Enter your Choice: 4

*******************************
You are in case 4
Enter the value of a and b: 5 3
The value of 5 to the power 3 is : 125

*******************************
Enter 1 for Factorial
Enter 2 for Sum of Series of natural number upto n terms
Enter 3 for Fibonacci
Enter 4 for Power of a and b
Enter your Choice: 5

*******************************
Exiting...
PS C:\Users\aadil\Desktop\lab>
```

Program 2

```c
#include <stdio.h>

void printfunc(int arr[], int n)
{
    for (int i = 0; i < n; i++)
    {
        printf("%d ", arr[i]);
    }
}
void BubbleSort(int arr[], int n)
{
    int i, j;
    for (i = 0; i < n - 1; i++)
    {
        printf("Iteration: %d\n", i + 1);

        for (j = 0; j < n - i - 1; j++)
        {
            if (arr[j] > arr[j + 1])
            {
                int temp = arr[j];
                arr[j] = arr[j + 1];
                arr[j + 1] = temp;
            }
            printfunc(arr, n);
            printf("\n");
        }
    }
}

void BubbleSortEarlyTermination(int arr[], int n)
{
    int i, j, f = 1;
    for (i = 0; i < n - 1; i++)
    {
        f = 1;
        printf("Iteration: %d\n", i + 1);
        for (j = 0; j < n - i - 1; j++)
        {
            if (arr[j] > arr[j + 1])
            {
                f = 0;
                int temp = arr[j];
                arr[j] = arr[j + 1];
```

```
                arr[j + 1] = temp;
            }
            printfunc(arr, n);
            printf("\n");
        }

        if (f == 1)
        {
            printf("Array is sorted!\n");
            break;
        }
    }
}

int main()
{
    int arr[100], n;

    scanf("%d", &n);
    for (int i = 0; i < n; i++)
    {
        scanf("%d", &arr[i]);
    }
    BubbleSort(arr, n);
    BubbleSortEarlyTermination(arr, n);

    return 0;
}
```

Output:

```
10
5
6
1
78
Iteration: 1
5 10 6 1 78
5 6 10 1 78
5 6 1 10 78
5 6 1 10 78
Iteration: 2
5 6 1 10 78
5 1 6 10 78
5 1 6 10 78
Iteration: 3
1 5 6 10 78
1 5 6 10 78
Iteration: 4
1 5 6 10 78
```

```
5
5
10
15
20
25
Iteration: 1
5 10 15 20 25
5 10 15 20 25
5 10 15 20 25
5 10 15 20 25
Array is sorted!
```

Program 3

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

struct employee
{
    int empid;
    char name[20];
    int salary;
};
struct employee e[10];
int count = 0;
void add(int n)
{
    for (int i = 0; i < n; i++)
    {
        printf("Employee id: ");
        scanf("%d", &e[i].empid);
        printf("Employee Name: ");
        scanf("%s", &e[i].name);
        printf("Employee Salary: ");
        scanf("%d", &e[i].salary);
        count++;
    }
    printf("Employees Added Successfully\n");
}
void display()
{

    if (count == 0)
    {
        printf("No Employee to Display\n");
    }
    else
    {
        for (int i = 0; i < count; i++)
        {
            printf("Employee No. %d\n", i + 1);
            printf("Id: %d\n", e[i].empid);
            printf("Name: %s\n", e[i].name);
            printf("Salary: %d\n", e[i].salary);
        }
    }
}
```

```c
void searchbyid(int id)
{
    if (count == 0)
    {
        printf("No Employee to Search\n");
    }
    else
    {
        int f = 0;
        for (int i = 0; i < count; i++)
        {
            if (e[i].empid == id)
            {
                f = 1;

                printf("Id: %d\n", e[i].empid);
                printf("Name: %s\n", e[i].name);
                printf("Salary: %d\n", e[i].salary);
            }
        }
        if (f == 0)
        {
            printf("No such employee found\n");
        }
    }
}
void searchbyname(char name[])
{
    if (count == 0)
    {
        printf("No Employee to Search\n");
    }
    else
    {
        int f = 0;
        for (int i = 0; i < count; i++)
        {
            if (strcmp(e[i].name, name) == 0)
            {
                f = 1;
                printf("Id: %d\n", e[i].empid);
                printf("Name: %s\n", e[i].name);
                printf("Salary: %d\n", e[i].salary);
            }
        }
        if (f == 0)
```

```
        {
            printf("No such employee found\n");
        }
    }
}
void highestsalary()
{
    int maxsalary = 0, id = 0;
    for (int i = 0; i < count; i++)
    {
        if (e[i].salary > maxsalary)
        {
            maxsalary = e[i].salary;
            id = e[i].empid;
        }
    }
    printf("Highest Salary is %d of the Employee having id %d\n", maxsalary,
id);
}
int main()
{

    while (1)
    {
        int ch;
        printf("Enter 1 to Add Employee\n");
        printf("Enter 2 to Display All Employee\n");
        printf("Enter 3 to Search Employee by empid\n");

        printf("Enter 4 to Search Employee by name\n");
        printf("Enter 5 to dispay Employee having highest Salary\n");
        printf("Enetr 6 to Exit\n");
        printf("Enter your Choice: ");
        scanf("%d", &ch);
        switch (ch)
        {
        case 1:
            printf("Case 1\n");
            printf("Enter the number of Employees you want to Add: ");
            int n;
            scanf("%d", &n);
            add(n);
            break;
        case 2:
            printf("Case 2\n");
            display();
```

```
            break;
        case 3:
            printf("Case 3\n");
            printf("Enter Employee id to Search: ");
            int id;
            scanf("%d", &id);
            searchbyid(id);
            break;
        case 4:
            printf("Case 4\n");
            printf("Enter Employee name to search: ");
            char name[20];

            scanf("%s", name);
            searchbyname(name);
            break;
        case 5:
            printf("Case 5\n");
            highestsalary();
            break;
        case 6:
            printf("Exiting\n");
            exit(0);
        }
    }

    return 0;
}
```

Output:

```
PS C:\Users\aadil\Desktop\CSE\lab> cd "c:\Users\aadil\Desktop\CSE\lab\" ; if ($?) { gcc structure.c -o structure } ; if ($?) { .\structure }
Enter 1 to Add Employee
Enter 2 to Display All Employee
Enter 3 to Search Employee by empid
Enter 4 to Search Employee by name
Enter 5 to dispay Employee having highest Salary
Enetr 6 to Exit
Enter your Choice: 1
Case 1
Enter the number of Employees you want to Add: 2
Employee id: 101
Employee Name: adil
Employee Salary: 10000
Employee id: 102
Employee Name: soban
Employee Salary: 15000
Employees Added Successfully
Enter 1 to Add Employee
Enter 2 to Display All Employee
Enter 3 to Search Employee by empid
Enter 4 to Search Employee by name
Enter 5 to dispay Employee having highest Salary
Enetr 6 to Exit
Enter your Choice: 2
Case 2
Employee No. 1
Id: 101
Name: adil
Salary: 10000
Employee No. 2
Id: 102
Name: soban
Salary: 15000
Enter 1 to Add Employee
Enter 2 to Display All Employee
Enter 3 to Search Employee by empid
Enter 4 to Search Employee by name
Enter 5 to dispay Employee having highest Salary
Enetr 6 to Exit
Enter your Choice: 3
```

```
Case 3
Enter Employee id to Search: 101
Id: 101
Name: adil
Salary: 10000
Enter 1 to Add Employee
Enter 2 to Display All Employee
Enter 3 to Search Employee by empid
Enter 4 to Search Employee by name
Enter 5 to dispay Employee having highest Salary
Enetr 6 to Exit
Enter your Choice: 4
Case 4
Enter Employee name to search: soban
Id: 102
Name: soban
Salary: 15000
Enter 1 to Add Employee
Enter 2 to Display All Employee
Enter 3 to Search Employee by empid
Enter 4 to Search Employee by name
Enter 5 to dispay Employee having highest Salary
Enetr 6 to Exit
Enter your Choice: 5
Case 5
Highest Salary is 15000 of the Employee having id 102
Enter 1 to Add Employee
Enter 2 to Display All Employee
Enter 3 to Search Employee by empid
Enter 4 to Search Employee by name
Enter 5 to dispay Employee having highest Salary
Enetr 6 to Exit
Enter your Choice: 6
Exiting
PS C:\Users\aadil\Desktop\CSE\lab>
```

Program 4

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

struct employee
{
    int empid;
    char name[20];
    int salary;
};
int count = 0, max;
void add(struct employee *e)
{
    if (count <= max)
    {
        printf("Employee id: ");
        scanf("%d", &(e + count)->empid);
        printf("Employee Name: ");
        scanf("%s", &(e + count)->name);
        printf("Employee Salary: ");
        scanf("%d", &(e + count)->salary);
        count++;
        printf("Employees Added Successfully\n");
    }
    else
    {
        printf("Exceeding Maximum Limit\n");
    }
}
void display(struct employee *e)
{
    if (count == 0)
    {
        printf("No Employee to Display\n");
    }
    else
    {
        printf("Employee Id | Name | Salary\n\n");
        for (int i = 0; i < count; i++)
        {

            printf(" %d %s %d\n", (e + i)->empid, (e + i)->name, (e + i)-
>salary);
        }
```

```
    }
}
void searchbyid(int id, struct employee *e)
{
    if (count == 0)
    {
        printf("No Employee to Search\n");
    }
    else
    {
        int f = 0;
        for (int i = 0; i < count; i++)
        {
            if ((e + i)->empid == id)

            {
                f = 1;
                printf("Id: %d\n", (e + i)->empid);
                printf("Name: %s\n", (e + i)->name);
                printf("Salary: %d\n", (e + i)->salary);
            }
        }
        if (f == 0)
        {
            printf("No such employee found\n");
        }
    }
}
void searchbyname(char name[], struct employee *e)
{
    if (count == 0)
    {
        printf("No Employee to Search\n");
    }
    else
    {
        int f = 0;
        for (int i = 0; i < count; i++)
        {
            if (strcmp((e + i)->name, name) == 0)
            {
                f = 1;
                printf("Id: %d\n", (e + i)->empid);
                printf("Name: %s\n", (e + i)->name);

                printf("Salary: %d\n", (e + i)->salary);
```

```
            }
        }
        if (f == 0)
        {
            printf("No such employee found\n");
        }
    }
}
void highestsalary(struct employee *e)
{
    int maxsalary = 0, id = 0;
    for (int i = 0; i < count; i++)
    {
        if ((e + i)->salary > maxsalary)
        {
            maxsalary = (e + i)->salary;
            id = (e + i)->empid;
        }
    }
    printf("Highest Salary is %d of the Employee having id %d\n", maxsalary,
id);
}
int main()
{
    printf("Maximum Number of Employees: ");
    scanf("%d", &max);
    struct employee *e;
    e = (struct employee *)malloc(max * sizeof(struct employee));

    while (1)
    {
        int ch;
        printf("Enter 1 to Add Employee\n");
        printf("Enter 2 to Display All Employee\n");
        printf("Enter 3 to Search Employee by empid\n");
        printf("Enter 4 to Search Employee by name\n");
        printf("Enter 5 to dispay Employee having highest Salary\n");
        printf("Enetr 6 to Exit\n");
        printf("Enter your Choice: ");
        scanf("%d", &ch);
        switch (ch)
        {
        case 1:
            printf("Case 1\n");
            add(e);
            break;
```

```c
        case 2:
            printf("Case 2\n");
            display(e);
            break;
        case 3:
            printf("Case 3\n");
            printf("Enter Employee id to Search: ");
            int id;
            scanf("%d", &id);
            searchbyid(id, e);
            break;
        case 4:

            printf("Case 4\n");
            printf("Enter Employee name to search: ");
            char name[20];
            scanf("%s", name);
            searchbyname(name, e);
            break;
        case 5:
            printf("Case 5\n");
            highestsalary(e);
            break;
        case 6:
            printf("Exiting\n");
            exit(0);
        }
    }

    return 0;
}
```

OUTPUT

```
PS C:\Users\aadil\Desktop\CSE\dsalab> cd "c:\Users\aadil\Desktop\CSE\dsalab\" ; if ($?) { gcc dma.c -o dma } ; if ($?) { .\dma }
Maximum Number of Employees: 5
Enter 1 to Add Employee
Enter 2 to Display All Employee
Enter 3 to Search Employee by empid
Enter 4 to Search Employee by name
Enter 5 to dispay Employee having highest Salary
Enetr 6 to Exit
Enter your Choice: 1
Case 1
Employee id: 101
Employee Name: Adil
Employee Salary: 5000
Employees Added Successfully
Enter 1 to Add Employee
Enter 2 to Display All Employee
Enter 3 to Search Employee by empid
Enter 4 to Search Employee by name
Enter 5 to dispay Employee having highest Salary
Enetr 6 to Exit
Enter your Choice: 1
Case 1
Employee id: 102
Employee Name: Atif
Employee Salary: 10000
Employees Added Successfully
Enter 1 to Add Employee
Enter 2 to Display All Employee
Enter 3 to Search Employee by empid
Enter 4 to Search Employee by name
Enter 5 to dispay Employee having highest Salary
Enetr 6 to Exit
```

```
Enter your Choice: 1
Case 1
Employee id: 103
Employee Name: Arbaz
Employee Salary: 15000
Employees Added Successfully
Enter 1 to Add Employee
Enter 2 to Display All Employee
Enter 3 to Search Employee by empid
Enter 4 to Search Employee by name
Enter 5 to dispay Employee having highest Salary
Enetr 6 to Exit
Enter your Choice: 1
Case 1
Employee id: 104
Employee Name: Abu
Employee Salary: 12000
Employees Added Successfully
Enter 1 to Add Employee
Enter 2 to Display All Employee
Enter 3 to Search Employee by empid
Enter 4 to Search Employee by name
Enter 5 to dispay Employee having highest Salary
Enetr 6 to Exit
Enter your Choice: 1
Case 1
Employee id: 105
Employee Name: xyz
Employee Salary: 16000
Employees Added Successfully
Enter 1 to Add Employee
Enter 2 to Display All Employee
Enter 3 to Search Employee by empid
Enter 4 to Search Employee by name
Enter 5 to dispay Employee having highest Salary
Enetr 6 to Exit
Enter your Choice: 2
Case 2
Employee Id  |   Name  |  Salary

   101         Adil      5000
   102         Atif      10000
   103         Arbaz      15000
   104         Abu      12000
   105         xyz      16000
Enter 1 to Add Employee
Enter 2 to Display All Employee
Enter 3 to Search Employee by empid
Enter 4 to Search Employee by name
Enter 5 to dispay Employee having highest Salary
Enetr 6 to Exit
Enter your Choice: 3
Case 3
Enter Employee id to Search: 102
Id: 102
Name: Atif
Salary: 10000
Enter 1 to Add Employee
Enter 2 to Display All Employee
Enter 3 to Search Employee by empid
Enter 4 to Search Employee by name
Enter 5 to dispay Employee having highest Salary
Enetr 6 to Exit
Enter your Choice: 4
Case 4
Enter Employee name to search: Abu
Id: 104
Name: Abu
Salary: 12000
Enter 1 to Add Employee
Enter 2 to Display All Employee
Enter 3 to Search Employee by empid
Enter 4 to Search Employee by name
Enter 5 to dispay Employee having highest Salary
Enetr 6 to Exit
Enter your Choice: 5
Case 5
Highest Salary is 16000 of the Employee having id 105
Enter 1 to Add Employee
Enter 2 to Display All Employee
Enter 3 to Search Employee by empid
Enter 4 to Search Employee by name
Enter 5 to dispay Employee having highest Salary
Enetr 6 to Exit
Enter your Choice: 6
Exiting
```

20BCS042 MOHD ADIL

PROGRAM:

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

struct employee
{
   int empid;
   char name[20];
   int salary;
   struct employee *next;
};
int count = 0;
struct employee *add(struct employee *list)
{
   struct employee *e = (struct employee *)malloc(sizeof(struct employee));
   printf("Employee id: ");
   scanf("%d", &(e)->empid);
   printf("Employee Name: ");
   scanf("%s", &(e)->name);
   printf("Employee Salary: ");
   scanf("%d", &(e)->salary);
   e->next = list;
   list = e;
   count++;
   printf("Employees Added Successfully\n");
   return list;
}
void display(struct employee *list)
{
   if (count == 0)
```

```
    {
        printf("No Employee to Display\n");
    }
    else
    {
        printf("ID\tNAME\tSALARY\n");
        while (list != NULL)
        {
            printf("%d\t%s\t%d\n", (list)->empid, (list)->name, (list)->salary);
            list=list->next;
        }
    }
}
void searchbyid(int id, struct employee *list)
{
    while (list != NULL)
        {
            if ((list)->empid == id)
            {
                printf("ID\tNAME\tSALARY\n");
                printf("%d\t%s\t%d\n", (list)->empid, (list)->name, (list)->salary);
                break;
            }
            list=list->next;
        }

        if (list==NULL)
        {
            printf("No such employee found\n");
        }
}
void searchbyname(char name[], struct employee *list)
{
```

```
    while (list != NULL)
      {
         if (strcmp((list)->name, name) == 0)
         {
            printf("ID\tNAME\tSALARY\n");
            printf("%d\t%s\t%d\n", (list)->empid, (list)->name, (list)->salary);
            break;
         }
         list=list->next;
      }

      if (list== NULL)
      {
         printf("No such employee found\n");
      }
}
void highestsalary(struct employee *list)
{
   int maxsalary = 0;
   struct employee *temp = NULL;
   while (list != NULL)
   {
      if (list->salary >= maxsalary)
      {
         maxsalary = list->salary;
         temp = list;
      }
      list = list->next;
   }
   if (temp == NULL)
      printf("Employee not Found\n");
   else
   {
```

```
        printf("ID\tNAME\tSALARY\n");
        printf("%d\t%s\t%d\n", (temp)->empid, (temp)->name, (temp)->salary);
    }

    printf("Highest Salary is %d of the Employee having id %d\n", maxsalary, temp->empid);
}
int main()
{
    struct employee *list = NULL;

    printf("Enter 1 to Add Employee\n");
    printf("Enter 2 to Display All Employee\n");
    printf("Enter 3 to Search Employee by empid\n");
    printf("Enter 4 to Search Employee by name\n");
    printf("Enter 5 to dispay Employee having highest Salary\n");
    printf("Enter 6 to display number of Employees\n");
    printf("Enter 7 to Exit\n");
    while (1)
    {
        int ch;
        printf("Enter your Choice: ");
        scanf("%d", &ch);
        switch (ch)
        {
        case 1:
            printf("Case 1\n");
            list = add(list);
            break;
        case 2:
            printf("Case 2\n");
            display(list);
            break;
        case 3:
```

```
            printf("Case 3\n");
            printf("Enter Employee id to Search: ");
            int id;
            scanf("%d", &id);
            searchbyid(id, list);
            break;
        case 4:
            printf("Case 4\n");
            printf("Enter Employee name to search: ");
            char name[20];
            scanf("%s", name);
            searchbyname(name, list);
            break;
        case 5:
            printf("Case 5\n");
            highestsalary(list);
            break;
        case 6:
            printf("Case 6\n");
            printf("Total Number of Employees -> %d\n", count);
            break;
        case 7:
            printf("Exiting\n");
            exit(0);
        }
    }

    return 0;
}
```

OUTPUT

```
PS C:\Users\aadil\Desktop\CSE\dsalab> cd "c:\Users\aadil\Desktop\CSE\dsalab\" ; if ($?) { gcc dma2.c -o dma2 }
Enter 1 to Add Employee
Enter 2 to Display All Employee
Enter 3 to Search Employee by empid
Enter 4 to Search Employee by name
Enter 5 to dispay Employee having highest Salary
Enter 6 to display number of Employees
Enter 7 to Exit
Enter your Choice: 1
Case 1
Employee id: 101
Employee Name: adil
Employee Salary: 1000
Employees Added Successfully
Enter your Choice: 1
Case 1
Employee id: 102
Employee Name: abu
Employee Salary: 2000
Employees Added Successfully
Enter your Choice: 1
Case 1
Employee id: 103
Employee Name: aka
Employee Salary: 3000
Employees Added Successfully
Enter your Choice: 2
Case 2
ID      NAME    SALARY
103     aka     3000
102     abu     2000
101     adil    1000
Enter your Choice: 3
Case 3
Enter Employee id to Search: 102
ID      NAME    SALARY
102     abu     2000
Enter your Choice: 4
Case 4
Enter Employee name to search: adil
ID      NAME    SALARY
101     adil    1000
Enter your Choice: 5
Case 5
ID      NAME    SALARY
103     aka     3000
Highest Salary is 3000 of the Employee having id 103
Enter your Choice: 6
Case 6
Total Number of Employees -> 3
Enter your Choice: 7
Exiting
PS C:\Users\aadil\Desktop\CSE\dsalab>
```

20BCS042 MOHD ADIL

PROGRAM 6.a

```
#include <stdio.h>
#include <stdlib.h>
int size, top = -1;
int *stack;
void push()
{
   if (top >= size - 1)
      printf("Stack Overflow\n");
   else
   {
      printf("Enter Element->");
      int num;
      scanf("%d", &num);
      stack[++top] = num;
   }
}
void pop()
{
   if (top < 0)
      printf("Stack underflow\n");
   else
      printf("Popped element is->%d\n", stack[top--]);
}
void display()
{
   if (top >= 0)
   {
      printf("Elements are:");
      for (int i = 0; i <= top; i++)
         printf(" %d", stack[i]);
```

```
        printf("\n");
    }
    else
        printf("Stack is Empty\n");
}
int isEmpty()
{
    if (top == -1)
    {
        printf("Stack is Empty!!\n");
        return 1;
    }
    else
    {
        printf("No!\n");
        return 0;
    }
}
int isFull()
{
    if (top == size - 1)
    {
        printf("Stack is Full!!\n");
        return 1;
    }
    else
        printf("No!\n");
        return 0;
}
int main()
{
    int choice;
    printf("Input Max-size->");
```

```
scanf("%d", &size);
stack = (int *)malloc(size * sizeof(int));
printf("\n1.Push element\n");
printf("2.Pop element\n");
printf("3.Display elements\n");
printf("4.Stack is empty?\n");
printf("5.Stack is full?\n");
printf("6.Total elements\n");
printf("7.Exit\n");
while (1)
{
   printf("Enter the choice: ");
   scanf("%d", &choice);
   switch (choice)
   {
   case 1:
      push();
      break;
   case 2:
      pop();
      break;
   case 3:
      display();
      break;
   case 4:
      isEmpty();
      break;
   case 5:
      isFull();
      break;
   case 6:
      printf("Total number of elements->%d\n", top+1);
      break;
```

```
        case 7:
            printf("Exiting...");
            exit(0);
            break;
        }
    }

    return 0;
}
```

OUTPUT:

```
PS C:\Users\aadil\Desktop\CSE\dsalab> cd "c:\Users\aadil\Desktop\CSE\dsalab\" ; if ($?) { gcc program6a.c -o program6a }
Input Max-size->5

1.Push element
2.Pop element
3.Display elements
4.Stack is empty?
5.Stack is full?
6.Total elements
7.Exit
Enter the choice: 1
Enter Element->1
Enter the choice: 1
Enter Element->2
Enter the choice: 1
Enter Element->3
Enter the choice: 1
Enter Element->4
Enter the choice: 1
Enter Element->5
Enter the choice: 3
Elements are: 1 2 3 4 5
Enter the choice: 2
Popped element is->5
Enter the choice: 3
Elements are: 1 2 3 4
Enter the choice: 4
No!
Enter the choice: 5
No!
Enter the choice: 6
Total number of elements->4
Enter the choice: 7
Exiting...
PS C:\Users\aadil\Desktop\CSE\dsalab>
```

20BCS042 MOHD ADIL

PROGRAM 6.b:

```
#include <stdio.h>
#include <stdlib.h>
struct arr
{
   int a;
   struct arr *next;
};

struct arr *top = NULL;
int count = -1;

struct arr *push()
{
   struct arr *temp = (struct arr *)malloc(sizeof(struct arr));
   if (temp == NULL)
      printf("Heap Overflow\n");
   else
   {
      printf("Enter Element->");
      scanf("%d", &temp->a);
      temp->next = top;
      top = temp;
      count++;
      return top;
   }
}
void pop()
{
   struct arr *temp;
   if (top == NULL)
```

```
    {
        printf("Stack Underflow\n");
    }
    else
    {
        temp = top;
        top = top->next;
        temp->next = NULL;
        printf("Popped element->%d\n",temp->a);
        free(temp);
        count--;
    }
}
void display()
{
    struct arr* temp;
    if (count == -1)
    {
        printf("\nStack Underflow\n");
    }
    else
    {
        temp=top;
        printf("Elements are:");
        while (temp != NULL)
        {
            printf(" %d", (temp)->a);
            temp = temp->next;
        }
        printf("\n");
    }
}
int isEmpty()
```

```
{
   return top == NULL;
}
int peek()
{
   if (!isEmpty())
      return top->a;
}
int main()
{
   int choice;
   printf("\n1.Push element\n");
   printf("2.Pop element\n");
   printf("3.IsEmpty?\n");
   printf("4.Top or Peek element\n");
   printf("5.Total elements\n");
   printf("6.Display elements\n");
   printf("7.Exit\n");
   while (1)
   {
      printf("Enter the choice: ");
      scanf("%d", &choice);
      switch (choice)
      {
      case 1:
         push();
         break;
      case 2:
         pop();
         break;
      case 3:
         printf("%d\n", isEmpty());
         break;
```

```
        case 4:
            printf("Top element is -> %d\n",peek());
            break;
        case 5:
            printf("Total number of elements->%d\n", count + 1);
            break;
        case 6:
            display();
            break;
        case 7:
            printf("Exiting...");
            exit(0);
            break;
        }
    }
    return 0;
}
```

OUTPUT:

```
PS C:\Users\aadil\Desktop\CSE\dsalab> cd "c:\Users\aadil\Desktop\CSE\dsalab\" ; if ($?) { gcc program6b.c -o program6b }

1.Push element
2.Pop element
3.IsEmpty?
4.Top or Peek element
5.Total elements
6.Display elements
7.Exit
Enter the choice: 1
Enter Element->1
Enter the choice: 1
Enter Element->2
Enter the choice: 1
Enter Element->3
Enter the choice: 1
Enter Element->4
Enter the choice: 1
Enter Element->5
Enter the choice: 6
Elements are: 5 4 3 2 1
Enter the choice: 2
Popped element->5
Enter the choice: 6
Elements are: 4 3 2 1
Enter the choice: 3
0
Enter the choice: 4
Top element is -> 4
Enter the choice: 5
Total number of elements->4
Enter the choice: 6
Elements are: 4 3 2 1
Enter the choice: 7
Exiting...
PS C:\Users\aadil\Desktop\CSE\dsalab>
```

20BCS042 MOHD ADIL:

PROGRAM 7

```
#include <stdio.h>
#include <stdlib.h>
struct Queue
{
   int size;
   int front;
   int rear;
   int *Q;
};
void create(struct Queue *q, int size)
{
   q->size = size;
   q->front = q->rear = -1;
   q->Q = (int *)malloc(q->size * sizeof(int));
}
void enqueue(struct Queue *q)
{
   if (q->rear == q->size - 1)
      printf("Queue is Full\n");
   else
   {
      printf("Enter Element : ");
      int n;
      scanf("%d", &n);
      q->rear++;
      q->Q[q->rear] = n;
   }
}
int dequeue(struct Queue *q)
{
```

```
    int x = -1;
    if (q->front == q->rear)
        printf("Queue is Empty\n");
    else
    {

        q->front++;
        x = q->Q[q->front];
    }
    return x;
}
int isEmpty(struct Queue *q)
{
    if (q->front == q->rear)
        printf("Queue is Empty\n");
    return 1;
    return 0;
}
int isFull(struct Queue *q)
{
    if (q->rear >= q->size - 1)
        printf("Queue is full!\n");
    return 1;
    return 0;
}
void Display(struct Queue q)
{
    int i;
    for (i = q.front + 1; i <= q.rear; i++)
        printf("%d ", q.Q[i]);
    printf("\n");
}
int main()
```

```
{
   struct Queue q;
   create(&q, 5);
   int choice;
   printf("\n1. Enqueue\n2. Dequeue\n3. Front and Rear Element \n4. Isempty\n5. Isfull\n6. Total no of element\n7. Display\n8. Exit\n");
   while (1)
   {
      printf("Enter the choice: ");
      scanf("%d", &choice);
      switch (choice)
      {
      case 1:
         enqueue(&q);
         break;
      case 2:
         printf("Dequeued element->%d\n",dequeue(&q));
         break;
      case 3:
         printf("Front Element->%d\n", q.Q[q.front + 1]);
         printf("Rear Element->%d\n", q.Q[q.rear]);
      case 4:
         isEmpty(&q);
         break;
      case 5:
         isFull(&q);
         break;
      case 6:
         printf("Total number of elements->%d\n", q.rear - q.front);
         break;
      case 7:
         Display(q);
         break;
```

```
        case 8:
            printf("Exiting...");
            exit(0);
            break;
        }
    }

    return 0;
}
```

OUTPUT:

```
PS C:\Users\aadil\Desktop\CSE\dsalab> cd "c:\Users\aadil\Desktop\CSE\dsalab\" ; if ($?) { gcc program7.c -o program7 }

1. Enqueue
2. Dequeue
3. Front and Rear Element
4. Isempty
5. Isfull
6. Total no of element
7. Display
8. Exit
Enter the choice: 4
Queue is Empty
Enter the choice: 1
Enter Element : 1
Enter the choice: 1
Enter Element : 2
Enter the choice: 1
Enter Element : 3
Enter the choice: 1
Enter Element : 4
Enter the choice: 1
Enter Element : 5
Enter the choice: 5
Queue is full!
Enter the choice: 2
Dequeued element->1
Enter the choice: 3
Front Element->2
Rear Element->5
Enter the choice: 6
Total number of elements->4
Enter the choice: 7
2 3 4 5
Enter the choice: 8
Exiting...
PS C:\Users\aadil\Desktop\CSE\dsalab>
```

20BCS042 MOHD ADIL

PROGRAM 8: CIRCULAR QUEUE USING ARRAY

```c
#include <stdio.h>
#include <stdlib.h>
struct Queue
{
    int size;
    int front;
    int rear;
    int *Q;
};
void create(struct Queue *q, int size)
{
    q->size = size;
    q->front = q->rear = 0;
    q->Q = (int *)malloc(q->size * sizeof(int));
}
void enqueue(struct Queue *q)
{
    if ((q->rear + 1) % q->size == q->front)
        printf("Queue is Full\n");
    else
    {
        q->rear = (q->rear + 1) % q->size;
        printf("Enter Element : ");
        int n;
        scanf("%d", &n);
        q->Q[q->rear] = n;
    }
}
int dequeue(struct Queue *q)
{
    int x = -1;
    if (q->front == q->rear)
        printf("Queue is Empty\n");
    else
    {
        q->front = (q->front + 1) % q->size;
        x = q->Q[q->front];
    }
```

```
    return x;
}
int isEmpty(struct Queue *q)
{
    if (q->front == q->rear)
    {
        printf("Queue is Empty\n");
        return 1;
    }
    return 0;
}

int isFull(struct Queue *q)
{
    if ((q->rear + 1) % q->size == q->front)
    {
        printf("Queue is Full\n");
        return 1;
    }

    return 0;
}
void Display(struct Queue q)
{
    if (q.front == q.rear)
    {
        printf("Queue is Empty\n");
    }
    else
    {
        int i = (q.front + 1) % q.size;
        do
        {
            printf("%d ", q.Q[i]);
            i = (i + 1) % q.size;
        } while (i != (q.rear + 1) % q.size);
    }

    printf("\n");
}
int main()
{
```

```
    struct Queue q;
    create(&q, 6);
    int no_of_elements;
    int choice;
    printf("\n1. Enqueue\n2. Dequeue\n3. Front and Rear Element
\n4. Isempty\n5. Isfull\n6. Total no of element\n7. Display\n8.
Exit\n");
    while (1)
    {
        printf("Enter the choice: ");
        scanf("%d", &choice);
        switch (choice)
        {
        case 1:
            enqueue(&q);
            break;
        case 2:
            printf("Dequeued element->%d\n", dequeue(&q));
            break;
        case 3:
            printf("Front Element->%d\n", q.Q[q.front + 1]);
            printf("Rear Element->%d\n", q.Q[q.rear]);
        case 4:
            isEmpty(&q);
            break;
        case 5:
            isFull(&q);
            break;
        case 6:
            no_of_elements = q.front > q.rear ? (q.size - q.front
+ q.rear) : (q.rear - q.front);
            printf("Total number of elements->%d\n",
no_of_elements);
            break;
        case 7:
            Display(q);
            break;
        case 8:
            printf("Exiting...");
            exit(0);
            break;
        }
```

```
    }

    return 0;
}
```

OUTPUT:

```
PS C:\Users\aadil\Desktop\CSE\dsalab> cd "c:\Users\aadil\Desktop\CSE\dsalab\" ; if ($?) { gcc circularqueue.c -o circularqueue }

1. Enqueue
2. Dequeue
3. Front and Rear Element
4. Isempty
5. Isfull
6. Total no of element
7. Display
8. Exit
Enter the choice: 4
Queue is Empty
Enter the choice: 1
Enter Element : 0
Enter the choice: 1
Enter Element : 1
Enter the choice: 1
Enter Element : 2
Enter the choice: 1
Enter Element : 3
Enter the choice: 4
Enter the choice: 1
Enter Element : 4
Enter the choice: 1
Queue is Full
Enter the choice: 7
0 1 2 3 4
Enter the choice: 2
Dequeued element -> 0
Enter the choice: 7
1 2 3 4
Enter the choice: 1
Enter Element : 5
Enter the choice: 3
Front Element -> 1
Rear Element -> 5
Enter the choice: 6
Total number of elements->5
Enter the choice: 7
1 2 3 4 5
Enter the choice: 8
Exiting...
```

20BCS042 MOHD ADIL

PROGRAM 9: Queue using LL

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

struct QueueNode
{
    int data;
    struct QueueNode *QueueNext;
} *front = NULL, *rear = NULL;

int count=0;
void enqueue()
{
    struct QueueNode *temp = malloc(sizeof(struct
QueueNode));
    if (temp == NULL)
        printf("Heap Overflow\n");
    else
    {
        printf("Enter the Number : ");
        int x;
        scanf("%d", &x);
        temp->data = x;
        temp->QueueNext = NULL;
        if (front == NULL)
        {
            front = temp;
            rear = temp;
        }
        else
        {
            rear->QueueNext = temp;
            rear = temp;
        }
        count++;
    }
```

```
}
int dequeue()
{
    int x = -1;
    struct QueueNode *temp;
    if (front == NULL)
    {
        printf("Queue is Empty \n");
    }
    else
    {
        x = front->data;
        temp = front;
        front = front->QueueNext;
        free(temp);
        count--;
        return x;
    }
}
int isEmpty()
{
    if (front == NULL)
        return 1;
    return 0;
}
void Display()
{
    struct QueueNode *temp;
    temp = front;
    printf("Queue -> ");
    while (temp)
    {
        printf("%d | ", temp->data);
        temp = temp->QueueNext;
    }
    printf("\n");
}
int main()
```

```
{
    int choice;
    printf("\n1. Enqueue\n2. Dequeue\n3. Front and Rear
Element \n4. Isempty\n5. Total no of element\n6. Display\n7.
Exit\n");
    while (1)
    {
        printf("Enter the choice: ");
        scanf("%d", &choice);
        switch (choice)
        {
        case 1:
            enqueue();
            Display();
            break;
        case 2:
            printf("Dequeued element -> %d\n", dequeue());
            Display();
            break;
        case 3:
            printf("Front Element -> %d\n", front->data);
            printf("Rear Element -> %d\n", rear->data);
            break;
        case 4:
            printf("%d\n", isEmpty());
            break;
        case 5:
            printf("Total number of elements -> %d\n",
count);
            break;
        case 6:
            Display();
            break;
        case 7:
            printf("Exiting...");
            exit(0);
            break;
        }
```

```
    }

    return 0;
}
```

OUTPUT:

```
PS C:\Users\aadil\Desktop\CSE\dsalab> cd "c:\Users\aadil\Desktop\CSE\dsalab\" ; if ($?) { gcc program9.c -o program9 }

1. Enqueue
2. Dequeue
3. Front and Rear Element
4. Isempty
5. Total no of element
6. Display
7. Exit
Enter the choice: 1
Enter the Number : 1
Queue -> 1 |
Enter the choice: 1
Enter the Number : 2
Queue -> 1 | 2 |
Enter the choice: 1
Enter the Number : 3
Queue -> 1 | 2 | 3 |
Enter the choice: 3
Front Element -> 1
Rear Element -> 3
Enter the choice: 2
Dequeued element -> 1
Queue -> 2 | 3 |
Enter the choice: 3
Front Element -> 2
Rear Element -> 3
Enter the choice: 4
0
Enter the choice: 5
Total number of elements -> 2
Enter the choice: 6
Queue -> 2 | 3 |
Enter the choice: 7
Exiting...
PS C:\Users\aadil\Desktop\CSE\dsalab>
```

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

struct PQueue
{
    char n[4];
    int pr;
    struct PQueue *next;
} *front = NULL, *rear = NULL;

int count = 0;

void enqueue()
{
    struct PQueue *temp = malloc(sizeof(struct PQueue));
    if (temp == NULL)
        printf("Heap Overflow\n");
    else
    {
        printf("Enter the String : ");
        scanf("%s",temp->n);
        printf("Priority : ");
        scanf("%d",&temp->pr);
        temp->next = NULL;
        if (front == NULL || temp->pr < front->pr)
        {
            temp->next = front;
            front = temp;
        }
        else
        {
            struct PQueue *p = front;
            while (p->next != NULL && p->next->pr < temp->pr)
                p = p->next;
            temp->next = p->next;
            p->next = temp;
        }
        count++;
    }
}
void dequeue()
{
    if (front == NULL)
        printf("Queue Underflow\n");
```

```
    else
    {
        struct PQueue *temp = front;
        front = front->next;
        printf("Deleted Element : %s\n",temp->n);
        free(temp);
        count--;
    }
}
void Display()
{
    if (front == NULL)
        printf("Queue is Empty\n");
    else
    {
        struct PQueue *temp = front;
        printf("String\tPriority\n");
        while (temp != NULL)
        {
            printf("%s\t%d\n",temp->n,temp->pr);
            temp = temp->next;
        }
    }
}
int isEmpty()
{
    if (front == NULL)
        return 1;
    else
        return 0;
}
int main()
{
    int choice;
    printf("\n1. Enqueue\n2. Dequeue\n3. Front and Rear Element \n4.
Isempty\n5. Total no of element\n6. Display\n7. Exit\n");
    while (1)
    {
        printf("Enter the choice: ");
        scanf("%d", &choice);
        getchar();
        switch (choice)
        {
        case 1:
            enqueue();
            Display();
```

```
            break;
        case 2:
            dequeue();
            Display();
            break;
        case 3:
            printf("Front Element -> %s\n",front->n);
            struct PQueue *temp = front;
            while (temp->next != NULL)
            {
                temp = temp->next;
            }
            rear = temp;
            printf("Rear Element -> %s\n",rear->n);
            break;
        case 4:
            printf("%d\n", isEmpty());
            break;
        case 5:
            printf("Total number of elements -> %d\n", count);
            break;
        case 6:
            Display();
            break;
        case 7:
            printf("Exiting...");
            exit(0);
            break;
        }
    }
    return 0;
}
```

OUTPUT:

```
1. Enqueue
2. Dequeue
3. Front and Rear Element
4. Isempty
5. Total no of element
6. Display
7. Exit
Enter the choice: 1
Enter the String : abc
Priority : 1
String  Priority
abc     1
Enter the choice: 1
Enter the String : aka
Priority : 2
String  Priority
abc     1
aka     2
Enter the choice: 1
Enter the String : xyz
Priority : 0
String  Priority
xyz     0
abc     1
aka     2
Enter the choice: 2
Deleted Element : xyz
String  Priority
abc     1
aka     2
```

```
String  Priority
abc     1
aka     2
xyz     5
Enter the choice: 1
Enter the String : wvx
Priority : 3
String  Priority
abc     1
aka     2
wvx     3
xyz     5
Enter the choice: 3
Front Element -> abc
Rear Element -> xyz
Enter the choice: 4
0
Enter the choice: 5
Total number of elements -> 4
Enter the choice: 6
String  Priority
abc     1
aka     2
wvx     3
xyz     5
Enter the choice: 7
Exiting...
PS C:\Users\aadil\Desktop\CSE\dsalab>
```

```
#include <iostream>
using namespace std;
struct node
{
    int data;
    node *next;
} *first = NULL, *last = NULL;

int count = 0;
void insertatbegin(int x)
{
    node *temp = new node;
    temp->data = x;
    temp->next = NULL;
    if (first == NULL)
    {
        first = temp;
        last = temp;
        count++;
    }
    else
    {
        temp->next = first;
        first = temp;
        count++;
    }
}
void insertatend(int x)
{
    node *temp = new node;
    temp->data = x;
    temp->next = NULL;
    if (first == NULL)
    {
        first = temp;
        last = temp;
        count++;
    }
    else
    {
        last->next = temp;
        last = temp;
        count++;
    }
}
```

```
void insertatposition(int x, int pos)
{
    if (pos == 1)
    {
        insertatbegin(x);
    }
    else if (pos == count + 1)
    {
        insertatend(x);
    }
    else if (pos <= count)
    {
        node *temp = new node;
        temp->data = x;
        temp->next = NULL;
        node *p = first;
        for (int i = 1; i < pos - 1; i++)
        {
            p = p->next;
        }
        temp->next = p->next;
        p->next = temp;
        count++;
    }
    else
    {
        cout << "\nInvalid position";
    }
}
int deletefrombegin()
{
    if (first == NULL)
    {
        cout << "\nList is empty";
        return -1;
    }
    else
    {
        node *temp = first;
        first = first->next;
        int x = temp->data;
        delete temp;
        count--;
        return x;
    }
}
```

```
int deletefromend()
{
    if (first == NULL)
    {
        cout << "\nList is empty";
        return -1;
    }
    else
    {
        node *temp = first;
        while (temp->next != last)
        {
            temp = temp->next;
        }
        int x = last->data;
        last = temp;
        delete last->next;
        last->next = NULL;
        count--;
        return x;
    }
}
void search(int x)
{
    node *p = first;
    int pos = 1;
    while (p != NULL)
    {
        if (p->data == x)
        {
            cout << "\nElement found at position " << pos;
            return;
        }
        p = p->next;
        pos++;
    }
    cout << "\nElement not found";
}
void display()
{
    node *p = first;
    if (first == NULL)
    {
        cout << "\nList is empty";
    }
```

```
    else
    {
        cout << "\nList is: ";
        while (p != NULL)
        {
            cout << p->data << " ";
            p = p->next;
        }
        cout << "\nfirst -> " << first->data << " last -> " << last->data;
    }
}
int main()
{
    int choice, x, pos;
    cout << "\n1. Insert at begin"
         << "\n2. Insert at end"
         << "\n3. Insert at position"
         << "\n4. Delete from begin"
         << "\n5. Delete from end"
         << "\n6. Total number of elements"
         << "\n7. Search"
         << "\n8. Display"
         << "\n9. Exit";
    while (1)
    {
        cout << "\nEnter your choice: ";
        cin >> choice;
        switch (choice)
        {
        case 1:
            cout << "\nEnter the element to be inserted: ";
            cin >> x;
            insertatbegin(x);
            display();
            break;
        case 2:
            cout << "\nEnter the element to be inserted: ";
            cin >> x;
            insertatend(x);
            display();
            break;
        case 3:
            cout << "\nEnter the element to be inserted: ";
            cin >> x;
            cout << "\nEnter the position: ";
            cin >> pos;
```

```
            insertatposition(x, pos);
            display();
            break;
        case 4:
            cout << "\nDeleted element is: " << deletefrombegin();
            display();
            break;
        case 5:
            cout << "\nDeleted element is: " << deletefromend();
            display();
            break;
        case 6:
            cout << "Total number of elements in the list is " << count <<
endl;
            break;
        case 7:
            cout << "\nEnter the element to be searched: ";
            cin >> x;
            search(x);
            break;
        case 8:
            display();
            break;
        case 9:
            return 0;
        default:
            cout << "\nInvalid choice";
        }
    }
    return 0;
}
```

OUTPUT:

```
PS C:\Users\aadil\Desktop\CSE\dsalab> cd "c:\Users\aadil\Desktop\CSE\dsalab\" ; if ($?) { g++ program11.cpp -o program11 }

1. Insert at begin
2. Insert at end
3. Insert at position
4. Delete from begin
5. Delete from end
6. Total number of elements
7. Search
8. Display
9. Exit
Enter your choice: 1

Enter the element to be inserted: 1

List is: 1
first -> 1 last -> 1
Enter your choice: 2

Enter the element to be inserted: 10

List is: 1 10
first -> 1 last -> 10
Enter your choice: 2

Enter the element to be inserted: 40

List is: 1 10 40
first -> 1 last -> 40
Enter your choice: 3

Enter the element to be inserted: 30

Enter the position: 3

List is: 1 10 30 40
first -> 1 last -> 40
```

```
List is: 1 10 30 40
first -> 1 last -> 40
Enter your choice: 4

Deleted element is: 1
List is: 10 30 40
first -> 10 last -> 40
Enter your choice: 5

Deleted element is: 40
List is: 10 30
first -> 10 last -> 30
Enter your choice: 6
Total number of elements in the list is 2

Enter your choice: 7

Enter the element to be searched: 30

Element found at position 2
Enter your choice: 8

List is: 10 30
first -> 10 last -> 30
Enter your choice: 9
PS C:\Users\aadil\Desktop\CSE\dsalab>
```

```
#include <iostream>
using namespace std;
struct node
{
    node *prev;
    int data;
    node *next;
};

void insertatbegin(node *&head, int data)
{
    node *temp = new node;
    temp->data = data;
    temp->next = head;
    temp->prev = NULL;
    if (head != NULL)
    {
        head->prev = temp;
    }
    head = temp;
}
void insertatend(node *&head, int data)
{
    node *temp = new node;
    temp->data = data;
    temp->next = NULL;
    temp->prev = NULL;
    if (head == NULL)
    {
        head = temp;
        return;
    }
    node *temp1 = head;
    while (temp1->next != NULL)
    {
        temp1 = temp1->next;
    }
    temp1->next = temp;
    temp->prev = temp1;
}
void insertatpos(node *&head, int data, int pos)
{
    node *temp = new node;
    temp->data = data;
```

```
    temp->next = NULL;
    temp->prev = NULL;
    if (head == NULL)
    {
        head = temp;
        return;
    }
    node *temp1 = head;
    int i = 1;
    while (i < pos - 1)
    {
        temp1 = temp1->next;
        i++;
    }
    temp->next = temp1->next;
    temp->prev = temp1;
    temp1->next->prev = temp;
    temp1->next = temp;
}
void deleteatbegin(node *&head)
{
    if (head == NULL)
    {
        cout << "List is empty" << endl;
        return;
    }
    node *temp = head;
    head = head->next;
    head->prev = NULL;
    delete temp;
}
void deleteatend(node *&head)
{
    if (head == NULL)
    {
        cout << "List is empty" << endl;
        return;
    }
    node *temp = head;
    while (temp->next != NULL)
    {
        temp = temp->next;
    }
    temp->prev->next = NULL;
    delete temp;
}
```

```
void reversedisplay(node *head)
{
    if (head == NULL)
    {
        cout << "List is empty" << endl;
        return;
    }
    node *temp = head;
    while (temp->next != NULL)
    {
        temp = temp->next;
    }
    cout<<"Null";
    while (temp != NULL)
    {
        cout <<" <---> " << temp->data;
        temp = temp->prev;
    }
    cout <<" <---> Null"<<endl;
}
int search(node *head, int data)
{
    if (head == NULL)
    {
        cout << "List is empty" << endl;
        return -1;
    }
    node *temp = head;
    int i = 1;
    while (temp != NULL)
    {
        if (temp->data == data)
        {
            return i;
        }
        temp = temp->next;
        i++;
    }
    return -1;
}
void display(node *head)
{
    if (head == NULL)
    {
        cout << "List is empty" << endl;
        return;
```

```
    }
    node *temp = head;
    cout<<"Null";
    while (temp != NULL)
    {
        cout <<" <---> " << temp->data;
        temp = temp->next;
    }
    cout <<" <---> Null"<<endl;
}

int main()
{
    node *head = NULL;
    cout << "\n1. Insert at begin\n2. Insert at end\n3. Insert at given
position\n4. Deletion at begin\n5. Deletion at end\n6. Reverse Display\n7.
Search\n8. Display\n9. Exit\n";
    int ch, data, pos;
    while (1)
    {
        cout << "Enter your choice: ";
        cin >> ch;
        switch (ch)
        {
        case 1:
            cout << "Enter the data: ";
            cin >> data;
            insertatbegin(head, data);
            display(head);
            break;
        case 2:
            cout << "Enter the data: ";
            cin >> data;
            insertatend(head, data);
            display(head);
            break;
        case 3:
            cout << "Enter the data: ";
            cin >> data;
            cout << "Enter the position: ";
            cin >> pos;
            insertatpos(head, data, pos);
            display(head);
            break;
        case 4:
            deleteatbegin(head);
```

```
            display(head);
            break;
        case 5:
            deleteatend(head);
            display(head);
            break;
        case 6:
            reversedisplay(head);
            break;
        case 7:
            cout << "Enter the data: ";
            cin >> data;
            pos = search(head, data);
            if (pos == -1)
            {
                cout << "Data not found" << endl;
            }
            else
            {
                cout << "Data found at position: " << pos << endl;
            }
            break;
        case 8:
            display(head);
            break;
        case 9:
            cout<<"Exiting..."<<endl;
            return 0;
        default:
            cout << "Invalid choice" << endl;
        }
    }
    return 0;
}
```

OUTPUT:

```
PS C:\Users\aadil\Desktop\CSE\dsalab> cd "c:\Users\aadil\Desktop\CSE\dsalab\" ; if ($?) { g++ program12.cpp

1. Insert at begin
2. Insert at end
3. Insert at given position
4. Deletion at begin
5. Deletion at end
6. Reverse Display
7. Search
8. Display
9. Exit
Enter your choice: 1
Enter the data: 1
Null <---> 1 <---> Null
Enter your choice: 2
Enter the data: 2
Null <---> 1 <---> 2 <---> Null
Enter your choice: 2
Enter the data: 4
Null <---> 1 <---> 2 <---> 4 <---> Null
Enter your choice: 3
Enter the data: 3
Enter the position: 3
Null <---> 1 <---> 2 <---> 3 <---> 4 <---> Null
Enter your choice: 4
Null <---> 2 <---> 3 <---> 4 <---> Null
Enter your choice: 5
Null <---> 2 <---> 3 <---> Null
Null <---> 1 <---> 2 <---> 3 <---> 4 <---> 5 <---> Null
Enter your choice: 6
Null <---> 5 <---> 4 <---> 3 <---> 2 <---> 1 <---> Null
Enter your choice: 7
Enter the data: 3
Data found at position: 3
Enter your choice: 9
Exiting...
PS C:\Users\aadil\Desktop\CSE\dsalab>
```

THANK YOU