JAMIA MILLIA ISLAMIA, NEW DELHI

DATA STRUCTURE LAB

NAME: FAIZAN CHOUDHARY

ROLL NO: 20BCS021

SUBJECT CODE: CEN 391

SEMESTER: 3rd

COURSE: B.TECH.(COMPUTER ENGG.)

DEPT: DEPT OF COMPUTER ENGG.

PROGRAM NO	DATE	PROGRAM	REMARKS
1	14/09/2021	Menu driven program	
2	21/09/2021	Bubble Sort on array	
3	21/09/2021	Bubble Sort (early termination)	
4	05/10/2021	Employee records	
5	12/10/2021	Employee records 1 (dynamic)	
6	26/10/2021	Employee records 2 (dynamic)	
7(a)	02/11/2021	Stack using array	
7(b)	02/11/2021	Stack using linked list	
8	09/11/2021	Queue using array	
9	16/11/2021	Circular Queue	
10	23/11/2021	Queue using linked list	
11	30/11/2021	Priority Queue using linked list	
12	07/12/2021	Singly Linked List	
13	14/12/2021	Doubly Linked List	

FAIZAN CHOUDHARY

20BCS021

DSA LAB

14th September 2021

CODE:

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

void fact ()
{
   cout<<"\nEnter the number whose factorial is to be found: ";
   cin>>n;
   int temp=n;
   long long f=1;
   while (n>0)
   {
      f*=n;
      n--;
   }
   cout<<temp<<"! = "<<f<<endl;
}

void sum ()
{
   long sum=0;
   cout<<"\nEnter the number upto which sum is to be found (natural number): ";
```

```
    cin>>n;
    /*  OR
    sum=((n)*(n+1))/2;
    */
    if (n>0)
    {
       for (int i=0; i<=n; i++)
        sum+=i;
    }
    cout<<"\nSum of natural numbers upto "<<n<<" are: "<<sum;
}

void fibonacci ()
{
    int f=0, f1=1, f2=1, j=1;
    cout<<"\nEnter the limit upto which Fibonacci series is to be printed: ";
    cin>>n;
    cout<<"Fibonacci series: "<<endl;
    do
    {
       j++;
       cout<<f<<" ";
       f1=f2;
       f2=f;
       f=f1+f2;
    } while(j<=n);
}

void power()
{
    long a,b, res=1;
```

```
    cout<<"\nEnter the values of a and b: ";
    cin>>a>>b;
    int temp=b;
    while (b>0)
    {
        res*=a;
        b--;
    }
    cout<<"\n"<<a<<" ^ "<<temp<<" = "<<res;
}

int main()
{
    int ch;
    while (1)
    {
        A:
        cout<<"\n\nMENU:\n1. Factorial of a given number. \n2. Sum of series of natural numbers. \n3. Fibonacci Series. \n4. Power of a raised to b.\n5. Exit. ";
        cout<<"\nEnter your choice: ";
        cin>>ch;
        switch (ch)
        {
            case 1: fact();
                    break;
            case 2: sum();
                    break;
            case 3: fibonacci();
                    break;
            case 4: power();
                    break;
```

```
        case 5: exit(0);
        default: cout<<"\nWrong choice, enter again! ";
                goto A;
      }
   }

   cout<<"\n\nFAIZAN CHOUDHARY\n20BCS021";
   return 0;
}
```

OUTPUT:

```
FAIZAN CHOUDHARY
20BCS021


MENU:
1. Factorial of a given number.
2. Sum of series of natural numbers.
3. Fibonacci Series.
4. Power of a raised to b.
5. Exit.
Enter your choice: 1

Enter the number whose factorial is to be found: 10
10! = 3628800
```

```
MENU:
1. Factorial of a given number.
2. Sum of series of natural numbers.
3. Fibonacci Series.
4. Power of a raised to b.
5. Exit.
Enter your choice: 2

Enter the number upto which sum is to be found (natural number): 100

Sum of natural numbers upto 100 are: 5050
```

```
MENU:
1. Factorial of a given number.
2. Sum of series of natural numbers.
3. Fibonacci Series.
4. Power of a raised to b.
5. Exit.
Enter your choice: 3

Enter the limit upto which Fibonacci series is to be printed: 10
Fibonacci series:
0 1 1 2 3 5 8 13 21 34
```

```
MENU:
1. Factorial of a given number.
2. Sum of series of natural numbers.
3. Fibonacci Series.
4. Power of a raised to b.
5. Exit.
Enter your choice: 4

Enter the values of a and b: 6
6

6 ^ 6 = 46656

MENU:
1. Factorial of a given number.
2. Sum of series of natural numbers.
3. Fibonacci Series.
4. Power of a raised to b.
5. Exit.
Enter your choice: 5
```

FAIZAN CHOUDHARY

20BCS021

DSA LAB

21st September 2021

CODE:

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

void display (int a[], int n)
{
    for (int i=0; i<n; i++)
     cout<<a[i]<<" ";
}

void BubbleSort (int *a, int n)
{
    int temp;
    for (int i=0; i<n-1; i++)
    {
        cout<<"\nPASS "<<i+1<<endl;
        for (int j=0; j<(n-i); j++)
        {
            if (a[j]>a[j+1])
            {
                temp=a[j];
                a[j]=a[j+1];
                a[j+1]=temp;
            }
            cout<<"\nIteration "<<j+1<<": ";
            display(a,n);
            cout<<endl;
        }
    }
}

void BubbleSort_earlyterminate (int *a, int n)
{
    int temp,f=1;
    for (int i=0; i<n-1; i++)
    {
        f=1;
        cout<<"\n\nPASS "<<i+1<<endl;
        for (int j=0; j<(n-i); j++)
        {
            if (a[j]>a[j+1])
```

```
            {
                f=0;
                temp=a[j];
                a[j]=a[j+1];
                a[j+1]=temp;
            }
            cout<<"\nIteration "<<j+1<<": ";
            display(a,n);
            cout<<endl;
        }
        if (f==1)                       //early termination bubble sort
        {
            cout<<"\nArray has been sorted with early termination of the loop...";
            break;
        }
    }
}

int main()
{
    cout<<"\nFAIZAN CHOUDHARY\n20BCS021\n\n";
    int n;
    while (1)
    {
        cout<<"\nEnter the size of the array (press 0 to exit): ";
        cin>>n;
        if (n==0)
         exit(0);
        int a[n];
        cout<<"\nEnter the elements of the array: ";
        for (int i=0; i<n; i++)
         cin>>a[i];
        int temp,f=1,ch;
        A:
        cout<<"\nEnter choice: \n1. Bubble sort. \n2. Bubble sort using Early termination.
 \n3. Exit.\n";
        cin>>ch;
        switch (ch)
        {
            case 1: BubbleSort(a,n);
                    break;
            case 2: BubbleSort_earlyterminate(a,n);
                    break;
            case 3: exit(0);
            default: cout<<"Wrong choice entered! Try again! ";
                     goto A;
        }
        cout<<"\nArray after sorting: ";
        display(a,n);
        cout<<endl;
    }

    return 0;
}
```

OUTPUT:

```
FAIZAN CHOUDHARY
20BCS021


Enter the size of the array (press 0 to exit): 5

Enter the elements of the array: 1
2
3
4
5

Enter choice:
1. Bubble sort.
2. Bubble sort using Early termination.
3. Exit.
1

PASS 1

Iteration 1: 1 2 3 4 5

Iteration 2: 1 2 3 4 5

Iteration 3: 1 2 3 4 5

Iteration 4: 1 2 3 4 5

Iteration 5: 1 2 3 4 5
```

```
PASS 2

Iteration 1: 1 2 3 4 5

Iteration 2: 1 2 3 4 5

Iteration 3: 1 2 3 4 5

Iteration 4: 1 2 3 4 5

PASS 3

Iteration 1: 1 2 3 4 5

Iteration 2: 1 2 3 4 5

Iteration 3: 1 2 3 4 5

PASS 4

Iteration 1: 1 2 3 4 5

Iteration 2: 1 2 3 4 5

Array after sorting: 1 2 3 4 5
```

Early Termination:

```
Enter the size of the array (press 0 to exit): 5

Enter the elements of the array: 1
2
3
5
4

Enter choice:
1. Bubble sort.
2. Bubble sort using Early termination.
3. Exit.
2


PASS 1

Iteration 1: 1 2 3 5 4

Iteration 2: 1 2 3 5 4

Iteration 3: 1 2 3 5 4

Iteration 4: 1 2 3 4 5

Iteration 5: 1 2 3 4 5
```

```
PASS 2

Iteration 1: 1 2 3 4 5

Iteration 2: 1 2 3 4 5

Iteration 3: 1 2 3 4 5

Iteration 4: 1 2 3 4 5

Array has been sorted with early termination of the loop...
Array after sorting: 1 2 3 4 5
```

FAIZAN CHOUDHARY

20BCS021

DSA LAB

5th October 2021

CODE: (code pasted in this format for readability)

```
#include <iostream>
#include <string.h>
using namespace std;

struct employee {
    int empid;
    char name[20];
    int salary;
}emp[10];

int count=-1;

void add();
int check_id(int id)
{
    for (int i=0; i<=count; i++)
    {
        if (id==emp[i].empid)
        {
            cout<<"\nID already in record! Enter again...\n";
            count--;
            return -1;
        }
        return 0;
    }
}
void add ()
{
    int id;
    count++;
    cout<<"\nEnter the details of the employee:\n";
    cout<<"Enter the employee id: ";
    cin>>id;
    if (check_id(id)==-1)
     add();
    else
    {
        emp[count].empid=id;
        cout<<"Enter the employee name: ";
        char g=getchar();                       //or cin.ignore();
```

```
        cin.getline(emp[count].name, 20);
        cout<<"Enter salary: ";
        cin>>emp[count].salary;
    }
}

void display()
{
    cout<<"\nEMPLOYEE DETAILS:\n\n";
    cout<<"Employee ID\t\tEmployee name\t\tSalary\n";
    for (int i=0; i<=count; i++)
    {
        cout<<emp[i].empid;
        cout<<"\t\t\t"<<emp[i].name;
        cout<<"\t\t\t"<<emp[i].salary<<endl;
    }
}

void search_empid(int key)
{
    int flag=0;
    for (int i=0; i<=count; i++)
    {
        if (emp[i].empid==key)
        {
            flag=1;
            cout<<"\nEmployee found in record!";
            cout<<"\nEMPLOYEE DETAILS:\n\n";
            cout<<"Employee ID\t\tEmployee name\t\tSalary\n";
            cout<<emp[i].empid;
            cout<<"\t\t\t"<<emp[i].name;
            cout<<"\t\t\t"<<emp[i].salary;
        }
        if (flag==1)
         break;
        else if (flag==0)
         cout<<"\nEmployee not found in record!";
    }
}

void search_name(char test[])
{
    int flag=0;
    for (int i=0; i<=count; i++)
    {
        if (strcmp(test, emp[i].name)==0)
        {
            flag=1;
            cout<<"\nEmployee found in record!";
            cout<<"\nEMPLOYEE DETAILS:\n\n";
            cout<<"Employee ID\t\tEmployee name\t\tSalary\n";
            cout<<emp[i].empid;
            cout<<"\t\t\t"<<emp[i].name;
            cout<<"\t"<<emp[i].salary;
```

```
        }
        if (flag==1)
         return;
    }
    cout<<"\nEmployee not found in record!";
}

void highest_salary()
{
    int mx=0, index;
    for (int i=0; i<count; i++)
    {
        if (emp[i].salary>mx)
        {
            mx=emp[i].salary;
            index=i;
        }
    }
    cout<<"Employee with the highest salary is:\n";
    cout<<"Employee ID\t\tEmployee name\t\tSalary\n";
    cout<<emp[index+1].empid;
    cout<<"\t\t\t"<<emp[index+1].name;
    cout<<"\t\t\t"<<emp[index+1].salary;
}

int main()
{
    cout<<"\nFAIZAN CHOUDHARY\n20BCS021";

    int ch, key;
    char test[20], g1;
    while (1)
    {
        A:
        cout<<"\n\nMENU\n1. Add employee.\n2. Display all employees.\n3. Search employee b
y empid.\n4. Search employee by name.\n5. Employee having Highest Salary.\n6. Exit\n";
        cin>>ch;
        switch (ch)
        {
            case 1: if (count==10)
                    {
                        cout<<"\nMaximum employee limit (10) reached!\n";
                        goto A;
                    }
                    else if (count<10)
                     add();
                    break;
            case 2: if (count==-1)
                    {
                        cout<<"\nRecord is empty, add some employee details first!\n";
                        goto A;
                    }
                    else
                     display();
```

```
                break;
        case 3: cout<<"\nEnter employee ID to be searched for: ";
                cin>>key;
                search_empid(key);
                break;
        case 4: cout<<"\nEnter employee name to be searched for (case-sensitive): ";
                g1=getchar();
                cin.getline(test, 20);
                search_name(test);
                break;
        case 5: highest_salary();
                break;
        case 6: exit(0);
        default: cout<<"\nWrong choice! Enter again...\n";
                 goto A;
      }
   }
   return 0;
}
```

OUTPUT:

```
FAIZAN CHOUDHARY
20BCS021

MENU
1. Add employee.
2. Display all employees.
3. Search employee by empid.
4. Search employee by name.
5. Employee having Highest Salary.
6. Exit
1

Enter the details of the employee:
Enter the employee id: 12
Enter the employee name: Rakesh Kumar
Enter salary: 23000
```

```
MENU
1. Add employee.
2. Display all employees.
3. Search employee by empid.
4. Search employee by name.
5. Employee having Highest Salary.
6. Exit
1

Enter the details of the employee:
Enter the employee id: 12

ID already in record! Enter again...
```

```
Enter the details of the employee:
Enter the employee id: 13
Enter the employee name: Ganesh Pawar
Enter salary: 23900


MENU
1. Add employee.
2. Display all employees.
3. Search employee by empid.
4. Search employee by name.
5. Employee having Highest Salary.
6. Exit
2

EMPLOYEE DETAILS:

Employee ID             Employee name           Salary
12                      Rakesh Kumar                    23000
13                      Ganesh Pawar                    23900
```

```
MENU
1. Add employee.
2. Display all employees.
3. Search employee by empid.
4. Search employee by name.
5. Employee having Highest Salary.
6. Exit
3

Enter employee ID to be searched for: 12

Employee found in record!
EMPLOYEE DETAILS:

Employee ID             Employee name           Salary
12                      Rakesh Kumar                    23000
```

```
MENU
1. Add employee.
2. Display all employees.
3. Search employee by empid.
4. Search employee by name.
5. Employee having Highest Salary.
6. Exit
4

Enter employee name to be searched for (case-sensitive): Gajanan Anand

Employee not found in record!
```

```
MENU
1. Add employee.
2. Display all employees.
3. Search employee by empid.
4. Search employee by name.
5. Employee having Highest Salary.
6. Exit
5
Employee with the highest salary is:
Employee ID             Employee name           Salary
13                      Ganesh Pawar                    23900
```

```
MENU
1. Add employee.
2. Display all employees.
3. Search employee by empid.
4. Search employee by name.
5. Employee having Highest Salary.
6. Exit
6
```

FAIZAN CHOUDHARY

20BCS021

DSA LAB

12th October 2021

CODE: (code pasted in this format for readability)

```
#include <iostream>
#include <string.h>
using namespace std;

const int LIMIT=20;

struct employee {
    int empid;
    char name[20];
    int salary;
};

// struct employee *ptr = new employee[LIMIT];
struct employee *ptr = (struct employee *) malloc (LIMIT * sizeof (employee));

int count=-1;

void add();
int check_id(int id)
{
    for (int i=0; i<=count; i++)
    {
        if (id==(ptr+i)->empid)
        {
            cout<<"\nID already in record! Enter again...\n";
            count--;
            return -1;
        }
        return 0;
    }
}
void add ()
{
    int id;
    count++;
    cout<<"\nEnter the details of the employee:\n";
    cout<<"Enter the employee id: ";
    cin>>id;
    if (check_id(id)==-1)
     add();
```

```
    else
    {
        (ptr+count)->empid=id;
        cout<<"Enter the employee name: ";
        char g=getchar();                       //or cin.ignore();
        cin.getline((ptr+count)->name, 20);
        cout<<"Enter salary: ";
        cin>>(ptr+count)->salary;
    }
}

void display()
{
    cout<<"\nEMPLOYEE DETAILS:\n\n";
    cout<<"Employee ID\t\tEmployee name\t\tSalary\n";
    for (int i=0; i<=count; i++)
    {
        cout<<(ptr+i)->empid;
        cout<<"\t\t\t"<<(ptr+i)->name;
        cout<<"\t\t\t"<<(ptr+i)->salary<<endl;
    }
}

void search_empid(int key)
{
    int flag=0;
    for (int i=0; i<=count; i++)
    {
        if ((ptr+i)->empid==key)
        {
            flag=1;
            cout<<"\nEmployee found in record!";
            cout<<"\nEMPLOYEE DETAILS:\n\n";
            cout<<"Employee ID\t\tEmployee name\t\tSalary\n";
            cout<<(ptr+i)->empid;
            cout<<"\t\t\t"<<(ptr+i)->name;
            cout<<"\t\t\t"<<(ptr+i)->salary;
            if (flag==1)
             break;
            else
             cout<<"\nEmployee not found in record!";
        }
        if (flag==0)
        {
            cout<<"\nEmployee not found in record!";
            break;
        }
    }
}

void search_name(char test[])
{
    int flag=0;
    for (int i=0; i<=count; i++)
```

```
    {
        if (strcmp(test, (ptr+i)->name)==0)
        {
            flag=1;
            cout<<"\nEmployee found in record!";
            cout<<"\nEMPLOYEE DETAILS:\n\n";
            cout<<"Employee ID\t\tEmployee name\t\tSalary\n";
            cout<<(ptr+i)->empid;
            cout<<"\t\t\t"<<(ptr+i)->name;
            cout<<"\t\t\t"<<(ptr+i)->salary;
            if (flag==1)
             break;
        }
        // if (flag==0)
        //  cout<<"\nEmployee not found in record!";
    }
    if (flag==0)
     cout<<"\nEmployee not found in record!";
}

void highest_salary()
{
    int mx=0, index;
    for (int i=0; i<count; i++)
    {
        if ((ptr+i)->salary>mx)
        {
            mx=(ptr+i)->salary;
            index=i;
        }
    }
    cout<<"Employee with the highest salary is:\n";
    cout<<"Employee ID\t\tEmployee name\t\tSalary\n";
    cout<<(ptr+index)->empid;
    cout<<"\t\t\t"<<(ptr+index)->name;
    cout<<"\t\t\t"<<(ptr+index)->salary;
}

int main()
{
    cout<<"\nFAIZAN CHOUDHARY\n20BCS021\n";

    int ch, key,n;
    char test[20], g1;

    cout<<"\nEnter the number of employees initially: ";
    cin>>n;
    // struct employee *ptr1 = new employee[n];
    struct employee *ptr1 =(struct employee *) malloc (n * sizeof (employee));
    ptr=ptr1;

    while (1)
    {
        A:
```

```
        cout<<"\n\nMENU\n1. Add employee.\n2. Display all employees.\n3. Search employee
by empid.\n4. Search employee by name.\n5. Employee having Highest Salary.\n6. Exit\n";
        cin>>ch;
        switch (ch)
        {
            case 1: if (count==20)
                    {
                        cout<<"\nMaximum employee limit (20) reached!\n";
                        goto A;
                    }
                    else if (count<20)
                     add();
                    break;
            case 2: if (count==-1)
                    {
                        cout<<"\nRecord is empty, add some employee details first!\n";
                        goto A;
                    }
                    else
                     display();
                    break;
            case 3: cout<<"\nEnter employee ID to be searched for: ";
                    cin>>key;
                    search_empid(key);
                    break;
            case 4: cout<<"\nEnter employee name to be searched for (case-sensitive): ";
                    g1=getchar();
                    cin.getline(test, 20);
                    search_name(test);
                    break;
            case 5: highest_salary();
                    break;
            case 6: exit(0);
            default: cout<<"\nWrong choice! Enter again...\n";
                     goto A;
        }
    }
    return 0;
}
```

OUTPUT:

```
FAIZAN CHOUDHARY
20BCS021

Enter the number of employees initially: 3


MENU
1. Add employee.
2. Display all employees.
3. Search employee by empid.
4. Search employee by name.
5. Employee having Highest Salary.
6. Exit
1

Enter the details of the employee:
Enter the employee id: 12
Enter the employee name: Rakesh Kumar
Enter salary: 23000
```

```
MENU
1. Add employee.
2. Display all employees.
3. Search employee by empid.
4. Search employee by name.
5. Employee having Highest Salary.
6. Exit
1

Enter the details of the employee:
Enter the employee id: 12

ID already in record! Enter again...
```

```
Enter the details of the employee:
Enter the employee id: 13
Enter the employee name: Ganesh Pawar
Enter salary: 23900


MENU
1. Add employee.
2. Display all employees.
3. Search employee by empid.
4. Search employee by name.
5. Employee having Highest Salary.
6. Exit
2

EMPLOYEE DETAILS:

Employee ID             Employee name           Salary
12                      Rakesh Kumar                    23000
13                      Ganesh Pawar                    23900
```

```
MENU
1. Add employee.
2. Display all employees.
3. Search employee by empid.
4. Search employee by name.
5. Employee having Highest Salary.
6. Exit
3

Enter employee ID to be searched for: 12

Employee found in record!
EMPLOYEE DETAILS:

Employee ID             Employee name           Salary
12                      Rakesh Kumar                    23000
```

```
MENU
1. Add employee.
2. Display all employees.
3. Search employee by empid.
4. Search employee by name.
5. Employee having Highest Salary.
6. Exit
3

Enter employee ID to be searched for: 14

Employee not found in record!
```

```
MENU
1. Add employee.
2. Display all employees.
3. Search employee by empid.
4. Search employee by name.
5. Employee having Highest Salary.
6. Exit
4

Enter employee name to be searched for (case-sensitive): Gajanan Anand

Employee not found in record!
```

```
MENU
1. Add employee.
2. Display all employees.
3. Search employee by empid.
4. Search employee by name.
5. Employee having Highest Salary.
6. Exit
4

Enter employee name to be searched for (case-sensitive): Rakesh Kumar

Employee found in record!
EMPLOYEE DETAILS:

Employee ID             Employee name           Salary
12                      Rakesh Kumar                    23000
```

```
MENU
1. Add employee.
2. Display all employees.
3. Search employee by empid.
4. Search employee by name.
5. Employee having Highest Salary.
6. Exit
5
Employee with the highest salary is:
Employee ID             Employee name           Salary
13                      Ganesh Pawar                    23900
```

```
MENU
1. Add employee.
2. Display all employees.
3. Search employee by empid.
4. Search employee by name.
5. Employee having Highest Salary.
6. Exit
6
```

FAIZAN CHOUDHARY

20BCS021

DSA LAB

26th October 2021

CODE: (code pasted in this format for readability)

```
#include <iostream>
#include <string.h>
using namespace std;

struct employee {
    int empid;
    char name[20];
    int salary;
    struct employee *next;
};
struct employee *ptr,*top=NULL,*p;

int isEmpty ()
{
    if (top==NULL)
     return 1;
    else
     return 0;
}

int size ()
{
    if (isEmpty()==1)
     return 0;
    else
    {
        int count=0;
        for (p=top; p!=NULL; p=p->next)
         count++;
        return count;
    }
}

void displayall()
{
    cout<<"\nEMPLOYEE DETAILS:\n\n";
    cout<<"Employee ID\t\tEmployee name\t\tSalary\n";
    p=top;
    while (p!=NULL)
    {
```

```
        cout<<p->empid;
        cout<<"\t\t\t"<<p->name;
        cout<<"\t\t\t"<<p->salary<<endl;
        p=p->next;
    }
}

void displayone(employee *p)
{
    cout<<"\nEMPLOYEE DETAILS:\n\n";
    cout<<"Employee ID\t\tEmployee name\t\tSalary\n";
    cout<<p->empid;
    cout<<"\t\t\t"<<p->name;
    cout<<"\t\t\t"<<p->salary;
}

void add();
int check_id(int id)
{
    p=top;
    while(p!=NULL)
    {
        if (id==p->empid)
        {
            cout<<"\nID already in record! Enter again...\n";
            return -1;
        }
        p=p->next;
        return 0;
    }
}
void add ()
{
    int id;
    cout<<"\nEnter the details of the employee:\n";
    cout<<"Enter the employee id: ";
    cin>>id;
    if (check_id(id)==-1)
     add();
    else
    {
        ptr=(struct employee *) malloc (sizeof(struct employee));
        if (ptr==NULL)
        {
            cout<<"\nMemory could not be allocated!\n";
            return;
        }
        ptr->empid=id;
        cout<<"Enter the employee name: ";
        char g=getchar();                       //or cin.ignore();
        cin.getline(ptr->name, 20);
        cout<<"Enter salary: ";
        cin>>ptr->salary;
        ptr->next=NULL;
```

```
        if (top==NULL)          //if the stack is empty initally, directly assign top as
ptr
         top=ptr;
        else
        {
            ptr->next=top;      //otherwise assign the value that top points, to ptr and
then update the top to hold the address of the new ptr
            top=ptr;
        }
    }
}

void search_empid(int key, int f=0)
{
    int flag=0;
    p=top;
    while (p!=NULL)
    {
        if ((p->empid)==key)
        {
            flag=1;
            if (f==0)
             cout<<"\nEmployee found in record!";
            displayone(p);
            break;
        }
        p=p->next;
    }
    if (flag==0)
     cout<<"\nEmployee not found in record!";
}

void search_name(char test[])
{
    int flag=0;
    p=top;
    while (p!=NULL)
    {
        if (strcmp(test, p->name)==0)
        {
            flag=1;
            cout<<"\nEmployee found in record!";
            displayone(p);
            break;
        }
        p=p->next;
    }
    if (flag==0)
     cout<<"\nEmployee not found in record!";
}

void highest_salary()
{
```

```
    int mx=0,id;
    p=top;
    while (p!=NULL)
    {
        if ((p->salary)>mx)
        {
            mx=p->salary;
            id=p->empid;
        }
        p=p->next;
    }
    cout<<"Employee with the highest salary is:\n";
    search_empid(id,1);
    // displayone(p);
}

int main()
{
    cout<<"\nFAIZAN CHOUDHARY\n20BCS021\n";

    int ch, key, n,r;
    char test[20];

    while (1)
    {
        A:
        cout<<"\n\nMENU\n1. Add employee.\n2. Display all employees.\n3. Search employee
by empid.\n4. Search employee by name.\n5. Employee having Highest Salary.\n6. Number of
employee records.\n7. Exit\n";
        cin>>ch;
        switch (ch)
        {
            case 1: add();
                    break;
            case 2: if (isEmpty()==1)
                    {
                        cout<<"\nRecord is empty, add some employee details first!\n";
                        goto A;
                    }
                    else
                     displayall();
                    break;
            case 3: if (isEmpty()==1)
                    {
                        cout<<"\nRecord is empty, add some employee details first!\n";
                        goto A;
                    }
                    cout<<"\nEnter employee ID to be searched for: ";
                    cin>>key;
                    search_empid(key);
                    break;
            case 4: if (isEmpty()==1)
                    {
                        cout<<"\nRecord is empty, add some employee details first!\n";
```

```
                    goto A;
                }
                cout<<"\nEnter employee name to be searched for (case-sensitive): ";
                getchar();
                cin.getline(test, 20);
                search_name(test);
                break;
        case 5: if (isEmpty()==1)
                {
                    cout<<"\nRecord is empty, add some employee details first!\n";
                    goto A;
                }
                highest_salary();
                break;
        case 6: r=size();
                cout<<"\nNumber of employee records: "<<r;
                break;
        case 7: exit(0);
        default: cout<<"\nWrong choice! Enter again...\n";
                 goto A;
    }
  }
  return 0;
}
```

OUTPUT:

```
FAIZAN CHOUDHARY
20BCS021


MENU
1. Add employee.
2. Display all employees.
3. Search employee by empid.
4. Search employee by name.
5. Employee having Highest Salary.
6. Number of employee records.
7. Exit
1

Enter the details of the employee:
Enter the employee id: 12
Enter the employee name: Rakesh Kumar
Enter salary: 23000
```

```
MENU
1. Add employee.
2. Display all employees.
3. Search employee by empid.
4. Search employee by name.
5. Employee having Highest Salary.
6. Number of employee records.
7. Exit
1

Enter the details of the employee:
Enter the employee id: 12

ID already in record! Enter again...
```

```
Enter the details of the employee:
Enter the employee id: 13
Enter the employee name: Ganesh Pawar
Enter salary: 23900


MENU
1. Add employee.
2. Display all employees.
3. Search employee by empid.
4. Search employee by name.
5. Employee having Highest Salary.
6. Number of employee records.
7. Exit
2

EMPLOYEE DETAILS:

Employee ID             Employee name           Salary
13                      Ganesh Pawar                    23900
12                      Rakesh Kumar                    23000
```

```
 MENU
 1. Add employee.
 2. Display all employees.
 3. Search employee by empid.
 4. Search employee by name.
 5. Employee having Highest Salary.
 6. Number of employee records.
 7. Exit
 3

 Enter employee ID to be searched for: 12

 Employee found in record!
 EMPLOYEE DETAILS:

 Employee ID             Employee name           Salary
 12                      Rakesh Kumar                    23000
```

```
MENU
1. Add employee.
2. Display all employees.
3. Search employee by empid.
4. Search employee by name.
5. Employee having Highest Salary.
6. Number of employee records.
7. Exit
3

Enter employee ID to be searched for: 15

Employee not found in record!
```

```
MENU
1. Add employee.
2. Display all employees.
3. Search employee by empid.
4. Search employee by name.
5. Employee having Highest Salary.
6. Number of employee records.
7. Exit
4

Enter employee name to be searched for (case-sensitive): Rakesh Kumar

Employee found in record!
EMPLOYEE DETAILS:

Employee ID             Employee name           Salary
12                      Rakesh Kumar                    23000
```

```
MENU
1. Add employee.
2. Display all employees.
3. Search employee by empid.
4. Search employee by name.
5. Employee having Highest Salary.
6. Number of employee records.
7. Exit
5
Employee with the highest salary is:

EMPLOYEE DETAILS:

Employee ID             Employee name           Salary
13                      Ganesh Pawar                    23900
```

```
MENU
1. Add employee.
2. Display all employees.
3. Search employee by empid.
4. Search employee by name.
5. Employee having Highest Salary.
6. Number of employee records.
7. Exit
6

Number of employee records: 2
```

```
MENU
1. Add employee.
2. Display all employees.
3. Search employee by empid.
4. Search employee by name.
5. Employee having Highest Salary.
6. Number of employee records.
7. Exit
7
```

FAIZAN CHOUDHARY

20BCS021

DSA LAB

2nd November 2021

CODE: (code pasted in this format for readability)

```
#include <iostream>
using namespace std;
const int LIMIT=20;
int top=-1;

int *stack = (int *) malloc (LIMIT * sizeof(int));

int isEmpty ()
{
    if (top==-1)
     return 1;
    else
     return 0;
}

int isFull ()
{
    if (top==(LIMIT-1))
     return 1;
    else
     return 0;
}

void display ()
{
    if (isEmpty()==1)
     cout<<"\nStack is empty! Nothing to display\n";
    else
    {
        cout<<endl<<stack[top]<<"  <--"<<endl;
        for (int i=top-1; i>=0; i--)
         cout<<stack[i]<<endl;
    }
}

int size ()
{
    if (isEmpty()==1)
     return 0;
    else
```

```
     return (top+1);
}

void peek ()
{
    if (isEmpty()==1)
     cout<<"\nStack is empty..."<<endl;
    else
     cout<<"\nTop element is: "<<stack[top]<<endl;
}

void push (int n)
{
    if (isFull()==1)
     cout<<"\nStack Overflow! Maximum limit reached..."<<endl;
    else
    {
        top++;
        stack[top]=n;
        display();
    }
}

void pop ()
{
    if (isEmpty()==1)
     cout<<"\nStack Underflow! Stack is empty..."<<endl;
    else
    {
        cout<<"\nPopping top element: "<<stack[top]<<endl;
        top--;
        display();
    }
}

int main()
{
    cout<<"\nFAIZAN CHOUDHARY\n20BCS021\n";

    int ch,n,num, *stack1;

    cout<<"\nEnter number of elements initially: ";
    cin>>num;
    stack1 = (int *) malloc (num * sizeof(int));
    if (stack1==NULL)
    {
        cout<<"\nMemory could not be allocated!";
        exit(1);
    }
    stack = stack1;

    while (true)
    {
        A:
```

```
        cout<<"\nMENU:\n1. Push into stack\n2. Pop element\n3. Peek top element\n4. Check
if stack is full\n5. Check if stack is empty\n6. Size of the stack\n7. Display stack\n8.
Exit\n";
        cin>>ch;
        switch (ch)
        {
        case 1: cout<<"Enter the element to be pushed: ";
                cin>>n;
                push(n);
                break;
        case 2: pop();
                break;
        case 3: peek();
                break;
        case 4: if (isFull()==1)
                 cout<<"\nStack is full!\n";
                else
                 cout<<"\nStack is not full.\n";
                break;
        case 5: if (isEmpty()==1)
                 cout<<"\nStack is empty!\n";
                else
                 cout<<"\nStack is not empty.\n";
                break;
        case 6: cout<<"\nSize of the stack is: "<<size()<<endl;
                break;
        case 7: cout<<"\nStack elements: "<<endl;
                display();
                break;
        case 8: exit(0);
        default: cout<<"\nWrong choice! Enter again...\n";
                 goto A;
        }
    }
}
```

OUTPUT:

```
FAIZAN CHOUDHARY
20BCS021

Enter number of elements initially: 5

MENU:
1. Push into stack
2. Pop element
3. Peek top element
4. Check if stack is full
5. Check if stack is empty
6. Size of the stack
7. Display stack
8. Exit
1
Enter the element to be pushed: 55

55  <--
```

```
MENU:
1. Push into stack
2. Pop element
3. Peek top element
4. Check if stack is full
5. Check if stack is empty
6. Size of the stack
7. Display stack
8. Exit
1
Enter the element to be pushed: 44

44  <--
55
```

```
MENU:
1. Push into stack
2. Pop element
3. Peek top element
4. Check if stack is full
5. Check if stack is empty
6. Size of the stack
7. Display stack
8. Exit
1
Enter the element to be pushed: 33

33  <--
44
55
```

```
MENU:
1. Push into stack
2. Pop element
3. Peek top element
4. Check if stack is full
5. Check if stack is empty
6. Size of the stack
7. Display stack
8. Exit
3

Top element is: 33
```

```
MENU:
1. Push into stack
2. Pop element
3. Peek top element
4. Check if stack is full
5. Check if stack is empty
6. Size of the stack
7. Display stack
8. Exit
4

Stack is not full.
```

```
MENU:
1. Push into stack
2. Pop element
3. Peek top element
4. Check if stack is full
5. Check if stack is empty
6. Size of the stack
7. Display stack
8. Exit
6

Size of the stack is: 3
```

```
MENU:
1. Push into stack
2. Pop element
3. Peek top element
4. Check if stack is full
5. Check if stack is empty
6. Size of the stack
7. Display stack
8. Exit
2

Popping top element: 33

44  <--
55
```

```
MENU:
1. Push into stack
2. Pop element
3. Peek top element
4. Check if stack is full
5. Check if stack is empty
6. Size of the stack
7. Display stack
8. Exit
2

Popping top element: 44

55  <--
```

```
MENU:
1. Push into stack
2. Pop element
3. Peek top element
4. Check if stack is full
5. Check if stack is empty
6. Size of the stack
7. Display stack
8. Exit
2

Popping top element: 55

Stack is empty! Nothing to display
```

```
MENU:
1. Push into stack
2. Pop element
3. Peek top element
4. Check if stack is full
5. Check if stack is empty
6. Size of the stack
7. Display stack
8. Exit
5

Stack is empty!
```

FAIZAN CHOUDHARY

20BCS021

DSA LAB

2nd November 2021

CODE: (code pasted in this format for readability)

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

struct stack
{
    int info;
    struct stack *next;
};
struct stack *ptr,*top=NULL,*p;

int isEmpty ()
{
    if (top==NULL)
     return 1;
    else
     return 0;
}

void display ()
{
    if (isEmpty()==1)
     cout<<"\nStack is empty! Nothing to display\n";
    else
    {
        cout<<endl<<top->info<<"  <--"<<endl;
        p=top->next;
        while (p!=NULL)
        {
            cout<<p->info<<endl;
            p=p->next;
        }
    }
}

int size ()
{
    if (isEmpty()==1)
     return 0;
    else
    {
```

```
        int count=0;
        for (p=top; p!=NULL; p=p->next)
         count++;
        return count;
    }
}

void peek ()
{
    if (isEmpty()==1)
     cout<<"\nStack is empty..."<<endl;
    else
     cout<<"\nTop element is: "<<top->info<<endl;
}

void push (int n)
{
    ptr=(struct stack *) malloc (sizeof(struct stack));
    if (ptr==NULL)
    {
        cout<<"\nMemory could not be allocated!\n";
        return;
    }
    ptr->info=n;
    ptr->next=NULL;         //assign values to the empty dynamic block
    if (top==NULL)          //if the stack is empty initally, directly assign top as ptr
     top=ptr;
    else
    {
        ptr->next=top;      //otherwise assign the value that top points, to ptr and then
update the top to hold the address of the new ptr
        top=ptr;
    }
    display();
}

void pop ()
{
    if (isEmpty()==1)
     cout<<"\nStack Underflow! Stack is empty..."<<endl;
    else
    {
        cout<<"\nPopping top element: "<<top->info<<endl;
        top=top->next;
        display();
    }
}

int main()
{
    cout<<"\nFAIZAN CHOUDHARY\n20BCS021\n";

    int ch,n;
    while (true)
```

```
    {
        A:
        cout<<"\nMENU:\n1. Push into stack\n2. Pop element\n3. Peek top element\n4. Check
if stack is empty\n5. Size of the stack\n6. Display stack\n7. Exit\n";
        cin>>ch;
        switch (ch)
        {
        case 1: cout<<"Enter the element to be pushed: ";
                cin>>n;
                push(n);
                break;
        case 2: pop();
                break;
        case 3: peek();
                break;
        case 4: if (isEmpty()==1)
                 cout<<"\nStack is empty!\n";
                else
                 cout<<"\nStack is not empty.\n";
                break;
        case 5: cout<<"\nSize of the stack is: "<<size()<<endl;
                break;
        case 6: cout<<"\nStack elements: "<<endl;
                display();
                break;
        case 7: exit(0);
        default: cout<<"\nWrong choice! Enter again...\n";
                 goto A;
        }
    }
}
```

OUTPUT:

```
FAIZAN CHOUDHARY
20BCS021

MENU:
1. Push into stack
2. Pop element
3. Peek top element
4. Check if stack is empty
5. Size of the stack
6. Display stack
7. Exit
1
Enter the element to be pushed: 45

45  <--
```

```
MENU:
1. Push into stack
2. Pop element
3. Peek top element
4. Check if stack is empty
5. Size of the stack
6. Display stack
7. Exit
1
Enter the element to be pushed: 43

43  <--
45
```

```
MENU:
1. Push into stack
2. Pop element
3. Peek top element
4. Check if stack is empty
5. Size of the stack
6. Display stack
7. Exit
1
Enter the element to be pushed: 76

76  <--
43
45
```

```
MENU:
1. Push into stack
2. Pop element
3. Peek top element
4. Check if stack is empty
5. Size of the stack
6. Display stack
7. Exit
3

Top element is: 76
```

```
MENU:
1. Push into stack
2. Pop element
3. Peek top element
4. Check if stack is empty
5. Size of the stack
6. Display stack
7. Exit
5

Size of the stack is: 3
```

```
MENU:
1. Push into stack
2. Pop element
3. Peek top element
4. Check if stack is empty
5. Size of the stack
6. Display stack
7. Exit
2

Popping top element: 76

43  <--
45
```

```
MENU:
1. Push into stack
2. Pop element
3. Peek top element
4. Check if stack is empty
5. Size of the stack
6. Display stack
7. Exit
2

Popping top element: 43

45  <--
```

```
MENU:
1. Push into stack
2. Pop element
3. Peek top element
4. Check if stack is empty
5. Size of the stack
6. Display stack
7. Exit
2

Popping top element: 45

Stack is empty! Nothing to display
```

```
MENU:
1. Push into stack
2. Pop element
3. Peek top element
4. Check if stack is empty
5. Size of the stack
6. Display stack
7. Exit
4

Stack is empty!
```

FAIZAN CHOUDHARY

20BCS021

DSA LAB

9th November 2021

CODE: (code pasted in this format for readability)

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

const int LIMIT=20;
int queue[LIMIT], front=-1, rear=-1;

int isEmpty ()
{
    if (front==-1 && rear==-1)
        return 1;
    else
        return 0;
}

int isFull ()
{
    if (rear==(LIMIT-1))
        return 1;
    else
        return 0;
}

void display ()
{
    if (isEmpty()==1)
        cout<<"\nQueue is empty! Nothing to display\n";
    else
    {
        for (int i=front; i<rear; i++)
            cout<<queue[i]<<" <- ";
        cout<<queue[rear]<<endl;
    }
}

int size ()
{
    if (isEmpty()==1)
        return 0;
    else
        return (rear-front+1);
```

```
}

void front_rear ()
{
    if (isEmpty()==1)
        cout<<"\nQueue is empty..."<<endl;
    else
    {
        cout<<"\nFront element is: "<<queue[front];
        cout<<"\nRear element is: "<<queue[rear]<<endl;
    }

}

void enqueue (int n)
{
    if (isFull()==1)
        cout<<"\nQueue Overflow! Maximum limit (20) reached..."<<endl;
    if (isEmpty()==1)
    {
        front=rear=0;
        queue[front]=n;
        display();
    }
    else
    {
        rear++;
        queue[rear]=n;
        display();
    }
}

void dequeue ()
{
    if (isEmpty()==1)
        cout<<"\nQueue Underflow! Queue is empty..."<<endl;
    if (front == rear)
    {
        cout<<"\nDequeuing front element: "<<queue[front]<<endl;
        front=rear=-1;
        display();
    }
    else
    {
        cout<<"\nDequeuing front element: "<<queue[front]<<endl;
        front++;
        display();
    }
}

int main()
{
    cout<<"\nFAIZAN CHOUDHARY\n20BCS021\n";
```

```
    int ch,n;
    while (true)
    {
        A:
        cout<<"\nMENU:\n1. Enqueue\n2. Dequeue\n3. Display front and rear elements\n4.
Check if queue is full\n5. Check if queue is empty\n6. Size of the queue\n7. Display
queue\n8. Exit\n";
        cin>>ch;
        switch (ch)
        {
        case 1: cout<<"\nEnter the element to be enqueued: ";
                cin>>n;
                enqueue(n);
                break;
        case 2: dequeue();
                break;
        case 3: front_rear();
                break;
        case 4: if (isFull()==1)
                    cout<<"\nQueue is full!\n";
                else
                    cout<<"\nQueue is not full.\n";
                break;
        case 5: if (isEmpty()==1)
                    cout<<"\nQueue is empty!\n";
                else
                    cout<<"\nQueue is not empty.\n";
                break;
        case 6: cout<<"\nSize of the queue is: "<<size()<<endl;
                break;
        case 7: cout<<"\nQueue elements: "<<endl;
                display();
                break;
        case 8: exit(0);
        default: cout<<"\nWrong choice! Enter again...\n";
                 goto A;
        }
    }
    return 0;
}
```

OUTPUT:

```
FAIZAN CHOUDHARY
20BCS021

MENU:
1. Enqueue
2. Dequeue
3. Display front and rear elements
4. Check if queue is full
5. Check if queue is empty
6. Size of the queue
7. Display queue
8. Exit
1

Enter the element to be enqueued: 22
22
```

```
MENU:
1. Enqueue
2. Dequeue
3. Display front and rear elements
4. Check if queue is full
5. Check if queue is empty
6. Size of the queue
7. Display queue
8. Exit
1

Enter the element to be enqueued: 33
22 <- 33
```

```
MENU:
1. Enqueue
2. Dequeue
3. Display front and rear elements
4. Check if queue is full
5. Check if queue is empty
6. Size of the queue
7. Display queue
8. Exit
1

Enter the element to be enqueued: 44
22 <- 33 <- 44
```

```
MENU:
1. Enqueue
2. Dequeue
3. Display front and rear elements
4. Check if queue is full
5. Check if queue is empty
6. Size of the queue
7. Display queue
8. Exit
3

Front element is: 22
Rear element is: 44
```

```
MENU:
1. Enqueue
2. Dequeue
3. Display front and rear elements
4. Check if queue is full
5. Check if queue is empty
6. Size of the queue
7. Display queue
8. Exit
6

Size of the queue is: 3
```

```
MENU:
1. Enqueue
2. Dequeue
3. Display front and rear elements
4. Check if queue is full
5. Check if queue is empty
6. Size of the queue
7. Display queue
8. Exit
4

Queue is not full.
```

```
MENU:
1. Enqueue
2. Dequeue
3. Display front and rear elements
4. Check if queue is full
5. Check if queue is empty
6. Size of the queue
7. Display queue
8. Exit
2

Dequeuing front element: 22
33 <- 44
```

```
MENU:
1. Enqueue
2. Dequeue
3. Display front and rear elements
4. Check if queue is full
5. Check if queue is empty
6. Size of the queue
7. Display queue
8. Exit
2

Dequeuing front element: 33
44
```

```
MENU:
1. Enqueue
2. Dequeue
3. Display front and rear elements
4. Check if queue is full
5. Check if queue is empty
6. Size of the queue
7. Display queue
8. Exit
2

Dequeuing front element: 44

Queue is empty! Nothing to display
```

```
MENU:
1. Enqueue
2. Dequeue
3. Display front and rear elements
4. Check if queue is full
5. Check if queue is empty
6. Size of the queue
7. Display queue
8. Exit
5

Queue is empty!
```

FAIZAN CHOUDHARY

20BCS021

DSA LAB

16th November 2021

CODE: (code pasted in this format for readability)

```
#include <iostream>
#include <limits.h>
using namespace std;

const int LIMIT=10;
int queue[LIMIT], front=-1, rear=-1;

int isEmpty ()
{
    if (front==-1 && rear==-1)
        return 1;
    else
        return 0;
}

int isFull ()
{
    if ((rear+1) % LIMIT == front)
        return 1;
    else
        return 0;
}

void display ()
{
    if (isEmpty()==1)
        cout<<"\nQueue is empty! Nothing to display\n";
    else
    {
        cout<<"\nQueue elements:\n";
        cout<<" <- ";
        for (int i=0; i<LIMIT; i++)
        {
            if (queue[i]==INT_MAX)
                cout<<"| - |";
            else
                cout<<"| "<<queue[i]<<" |";
        }
        cout<<" -> ";
    }
```

```
    cout<<"\nFront: "<<front<<"\tRear: "<<rear<<endl;
}

void front_rear ()
{
    if (isEmpty()==1)
        cout<<"\nQueue is empty..."<<endl;
    else
    {
        cout<<"\nFront element is: "<<queue[front];
        cout<<"\nRear element is: "<<queue[rear]<<endl;
    }

}

int size ()
{
    if (isEmpty()==1)
        return 0;
    else
        return (rear>front) ? (rear-front+1) : (front-rear+1);
}

void enqueue (int n)
{
    if (isFull()==1)
    {
        cout<<"\nQueue Overflow! Maximum limit (10) reached..."<<endl;
        return;
    }
    if (isEmpty()==1)
    {
        front=rear=0;
        queue[front]=n;
        display();
    }
    else
    {
        rear = (rear+1) % LIMIT;
        queue[rear]=n;
        display();
    }
}

void dequeue ()
{
    if (isEmpty()==1)
    {
        cout<<"\nQueue Underflow! Queue is empty..."<<endl;
        return;
    }
    if (front == rear)
    {
        cout<<"\nDequeuing front element: "<<queue[front]<<endl;
```

```
        queue[front] = INT_MAX;
        front=rear=-1;
        display();
    }
    else
    {
        cout<<"\nDequeuing front element: "<<queue[front]<<endl;
        queue[front] = INT_MAX;
        front++;
        display();
    }
}

int main()
{
    cout<<"\nFAIZAN CHOUDHARY\n20BCS021\n";

    int ch,n;

    // initialising with default values
    for (int i=0; i<LIMIT; i++)
        queue[i] = INT_MAX;

    while (true)
    {
        A:
        cout<<"\nMENU:\n1. Enqueue\n2. Dequeue\n3. Display front and rear elements\n4.
Check if queue is full\n5. Check if queue is empty\n6. Size of the queue\n7. Display
queue\n8. Exit\n";
        cin>>ch;
        switch (ch)
        {
        case 1: cout<<"\nEnter the element to be enqueued: ";
                cin>>n;
                enqueue(n);
                break;
        case 2: dequeue();
                break;
        case 3: front_rear();
                break;
        case 4: if (isFull()==1)
                    cout<<"\nQueue is full!\n";
                else
                    cout<<"\nQueue is not full.\n";
                break;
        case 5: if (isEmpty()==1)
                    cout<<"\nQueue is empty!\n";
                else
                    cout<<"\nQueue is not empty.\n";
                break;
        case 6: cout<<"\nSize of the queue is: "<<size()<<endl;
                break;
        case 7: cout<<"\nQueue elements: "<<endl;
                display();
```

```
                break;
        case 8: exit(0);
        default: cout<<"\nWrong choice! Enter again...\n";
                 goto A;
        }
    }
    return 0;
}
```

OUTPUT:

```
FAIZAN CHOUDHARY
20BCS021

MENU:
1. Enqueue
2. Dequeue
3. Display front and rear elements
4. Check if queue is full
5. Check if queue is empty
6. Size of the queue
7. Display queue
8. Exit
1

Enter the element to be enqueued: 22

Queue elements:
 <- | 22 || - || - || - || - || - || - || - || - | ->
Front: 0        Rear: 0
```

```
MENU:
1. Enqueue
2. Dequeue
3. Display front and rear elements
4. Check if queue is full
5. Check if queue is empty
6. Size of the queue
7. Display queue
8. Exit
1

Enter the element to be enqueued: 33

Queue elements:
 <- | 22 || 33 || - || - || - || - || - || - || - | ->
Front: 0        Rear: 1
```

```
MENU:
1. Enqueue
2. Dequeue
3. Display front and rear elements
4. Check if queue is full
5. Check if queue is empty
6. Size of the queue
7. Display queue
8. Exit
1

Enter the element to be enqueued: 44

Queue elements:
 <- | 22 || 33 || 44 || - || - || - || - || - || - | ->
Front: 0        Rear: 2
```

```
MENU:
1. Enqueue
2. Dequeue
3. Display front and rear elements
4. Check if queue is full
5. Check if queue is empty
6. Size of the queue
7. Display queue
8. Exit
3

Front element is: 22
Rear element is: 44
```

```
MENU:
1. Enqueue
2. Dequeue
3. Display front and rear elements
4. Check if queue is full
5. Check if queue is empty
6. Size of the queue
7. Display queue
8. Exit
6

Size of the queue is: 3
```

```
MENU:
1. Enqueue
2. Dequeue
3. Display front and rear elements
4. Check if queue is full
5. Check if queue is empty
6. Size of the queue
7. Display queue
8. Exit
4

Queue is not full.
```

```
MENU:
1. Enqueue
2. Dequeue
3. Display front and rear elements
4. Check if queue is full
5. Check if queue is empty
6. Size of the queue
7. Display queue
8. Exit
2

Dequeuing front element: 22

Queue elements:
 <- | - || 33 || 44 || - || - || - || - || - || - | ->
Front: 1        Rear: 2
```

```
MENU:
1. Enqueue
2. Dequeue
3. Display front and rear elements
4. Check if queue is full
5. Check if queue is empty
6. Size of the queue
7. Display queue
8. Exit
2

Dequeuing front element: 33

Queue elements:
 <- | - || - || 44 || - || - || - || - || - || - | ->
Front: 2        Rear: 2
```

```
MENU:
1. Enqueue
2. Dequeue
3. Display front and rear elements
4. Check if queue is full
5. Check if queue is empty
6. Size of the queue
7. Display queue
8. Exit
2

Dequeuing front element: 44

Queue is empty! Nothing to display

Front: -1       Rear: -1
```

```
MENU:
1. Enqueue
2. Dequeue
3. Display front and rear elements
4. Check if queue is full
5. Check if queue is empty
6. Size of the queue
7. Display queue
8. Exit
5

Queue is empty!
```

FAIZAN CHOUDHARY

20BCS021

DSA LAB

23rd November 2021

CODE: (code pasted in this format for readability)

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

struct queue
{
    int info;
    struct queue *next;
};
struct queue *ptr, *front=NULL, *rear=NULL, *p;

int isEmpty ()
{
    if (front==NULL || rear==NULL)
     return 1;
    else
     return 0;
}

void display ()
{
    if (isEmpty()==1)
     cout<<"\nQueue is empty! Nothing to display\n";
    else
    {
        p=front;
        while (p!=rear)
        {
            cout<<p->info<<" <- ";
            p=p->next;
        }
        cout<<rear->info<<endl;
    }
}

int size ()
{
    if (isEmpty()==1)
     return 0;
    else
    {
        int count=1;
        for (p=front; p!=rear; p=p->next)
```

```
         count++;
        return count;
    }
}

void front_rear ()
{
    if (isEmpty()==1)
     cout<<"\nQueue is empty..."<<endl;
    else
    {
        cout<<"\nFront element is: "<<front->info;
        cout<<"\nRear element is: "<<rear->info<<endl;
    }

}

void enqueue (int n)
{
    ptr=(struct queue *) malloc (sizeof(struct queue));
    if (ptr==NULL)
    {
        cout<<"\nMemory could not be allocated!\n";
        return;
    }
    if (front==NULL)
    {
        front=rear=ptr;
    }
    else
    {
        rear->next=ptr;
        rear=ptr;
    }
    ptr->info=n;
    ptr->next=NULL;
    display();
}

void dequeue ()
{
    if (isEmpty()==1)
     cout<<"\nQueue Underflow! Stack is empty..."<<endl;
    else
    {
        cout<<"\nDequeueing front element: "<<front->info<<endl;
        front=front->next;
        display();
    }
}

int main()
{
    cout<<"\nFAIZAN CHOUDHARY\n20BCS021\n";
```

```
    int ch,n;

    while (true)
    {
        A:
        cout<<"\nMENU:\n1. Enqueue\n2. Dequeue\n3. Display front and rear elements\n4.
Check if queue is empty\n5. Size of the queue\n6. Display queue\n7. Exit\n";
        cin>>ch;
        switch (ch)
        {
        case 1: cout<<"\nEnter the element to be enqueued: ";
                cin>>n;
                enqueue(n);
                break;
        case 2: dequeue();
                break;
        case 3: front_rear();
                break;
        case 4: if (isEmpty()==1)
                 cout<<"\nQueue is empty!\n";
                else
                 cout<<"\nQueue is not empty.\n";
                break;
        case 5: cout<<"\nSize of the queue is: "<<size()<<endl;
                break;
        case 6: cout<<"\nQueue elements: "<<endl;
                display();
                break;
        case 7: exit(0);
        default: cout<<"\nWrong choice! Enter again...\n";
                 goto A;
        }
    }
}
```

OUTPUT:

```
FAIZAN CHOUDHARY
20BCS021

MENU:
1. Enqueue
2. Dequeue
3. Display front and rear elements
4. Check if queue is empty
5. Size of the queue
6. Display queue
7. Exit
1

Enter the element to be enqueued: 22
22
```

```
MENU:
1. Enqueue
2. Dequeue
3. Display front and rear elements
4. Check if queue is empty
5. Size of the queue
6. Display queue
7. Exit
1

Enter the element to be enqueued: 33
22 <- 33
```

```
MENU:
1. Enqueue
2. Dequeue
3. Display front and rear elements
4. Check if queue is empty
5. Size of the queue
6. Display queue
7. Exit
1

Enter the element to be enqueued: 44
22 <- 33 <- 44
```

```
MENU:
1. Enqueue
2. Dequeue
3. Display front and rear elements
4. Check if queue is empty
5. Size of the queue
6. Display queue
7. Exit
2

Dequeueing front element: 33
44
```

```
MENU:
1. Enqueue
2. Dequeue
3. Display front and rear elements
4. Check if queue is empty
5. Size of the queue
6. Display queue
7. Exit
3

Front element is: 22
Rear element is: 44
```

```
MENU:
1. Enqueue
2. Dequeue
3. Display front and rear elements
4. Check if queue is empty
5. Size of the queue
6. Display queue
7. Exit
2

Dequeueing front element: 44

Queue is empty! Nothing to display
```

```
MENU:
1. Enqueue
2. Dequeue
3. Display front and rear elements
4. Check if queue is empty
5. Size of the queue
6. Display queue
7. Exit
5

Size of the queue is: 3
```

```
MENU:
1. Enqueue
2. Dequeue
3. Display front and rear elements
4. Check if queue is empty
5. Size of the queue
6. Display queue
7. Exit
6

Queue elements:

Queue is empty! Nothing to display
```

```
MENU:
1. Enqueue
2. Dequeue
3. Display front and rear elements
4. Check if queue is empty
5. Size of the queue
6. Display queue
7. Exit
2

Dequeueing front element: 22
33 <- 44
```

```
MENU:
1. Enqueue
2. Dequeue
3. Display front and rear elements
4. Check if queue is empty
5. Size of the queue
6. Display queue
7. Exit
7
```

FAIZAN CHOUDHARY

20BCS021

DSA LAB

30th November 2021

CODE: (code pasted in this format for readability)

```
#include <iostream>
#include <string.h>
using namespace std;

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

int isEmpty ()
{
    if (front==NULL)
        return 1;
    else
        return 0;
}

void display ()
{
    if (isEmpty()==1)
    {
        cout<<"\nPriority Queue is empty! Nothing to display\n";
        return;
    }

    else
    {
        p=front;
        cout<<endl;
        while (p->next!=NULL)
        {
            cout<<"|| "<<p->n<<" | "<<p->pr<<" || --> ";
            p=p->next;
        }
        cout<<"|| "<<p->n<<" | "<<p->pr<<" || --> NULL"<<endl;
    }
}

int size ()
```

```
{
    int count=1;
    if (isEmpty()==1)
        return 0;
    else
    {
        p=front;
        while (p->next!=NULL)
        {
            count++;
            p=p->next;
        }
    }
    return count;
}

void front_rear ()
{

    if (isEmpty()==1)
    {
        cout<<"\nPriority Queue is empty..."<<endl;
        return;
    }
    p=front;
    while (p->next != NULL)
        p=p->next;
    rear=p;
    cout<<"\nFront element is: || "<<front->n<<" | "<<front->pr<<" ||";
    cout<<"\nRear element is: || "<<rear->n<<" | "<<rear->pr<<" ||"<<endl;
}

void enqueue (char* n, int pr)
{
    ptr=(struct PQueue *) malloc (sizeof(struct PQueue));
    if (ptr==NULL)
    {
        cout<<"\nMemory could not be allocated!\n";
        return;
    }
    strcpy(ptr->n, n);
    ptr->pr = pr;
    ptr->next=NULL;
    if (front==NULL || pr < (front->pr))
    {
        ptr->next = front;
        front=ptr;
    }
    else
    {
        p=front;
        while (p->next != NULL && p->next->pr <= pr)
            p=p->next;
        ptr->next = p->next;
```

```
        p->next = ptr;
    }

    display();
}

void dequeue ()
{
    if (isEmpty()==1)
        cout<<"\nPriority Queue Underflow!"<<endl;
    else
    {
        p = front;
        cout<<"\nDequeueing front element: || "<<p->n<<" | "<<p->pr<<" ||"<<endl;
        front=front->next;
        delete p;
        display();
    }
}

int main()
{
    cout<<"\nFAIZAN CHOUDHARY\n20BCS021\n";
    int ch,pr;
    char n[4];

    while (true)
    {
        A:
        cout<<"\nMENU:\n1. Enqueue\n2. Dequeue\n3. Display front and rear elements\n4.
Check if priority queue is empty\n5. Size of the priority queue\n6. Display priority
queue\n7. Exit\n";
        cin>>ch;
        switch (ch)
        {
            case 1: cout<<"\nEnter the element to be enqueued: ";
                    cin>>n;
                    cout<<"\nEnter the priority: ";
                    cin>>pr;
                    enqueue(n,pr);
                    break;
            case 2: dequeue();
                    break;
            case 3: front_rear();
                    break;
            case 4: if (isEmpty()==1)
                        cout<<"\nPriority Queue is empty!\n";
                    else
                        cout<<"\nPriority Queue is not empty.\n";
                    break;
            case 5: cout<<"\nSize of the priority queue is: "<<size()<<endl;
                    break;
            case 6: cout<<"\nPriority Queue elements: "<<endl;
                    display();
```

```
                    break;
            case 7: exit(0);
            default: cout<<"\nWrong choice! Enter again...\n";
                     goto A;
        }
    }
}
```

OUTPUT:

```
FAIZAN CHOUDHARY
20BCS021

MENU:
1. Enqueue
2. Dequeue
3. Display front and rear elements
4. Check if priority queue is empty
5. Size of the priority queue
6. Display priority queue
7. Exit
1

Enter the element to be enqueued: abc

Enter the priority: 3

|| abc | 3 || --> NULL
```

```
MENU:
1. Enqueue
2. Dequeue
3. Display front and rear elements
4. Check if priority queue is empty
5. Size of the priority queue
6. Display priority queue
7. Exit
1

Enter the element to be enqueued: bcd

Enter the priority: 1

|| bcd | 1 || --> || abc | 3 || --> NULL
```

```
MENU:
1. Enqueue
2. Dequeue
3. Display front and rear elements
4. Check if priority queue is empty
5. Size of the priority queue
6. Display priority queue
7. Exit
1

Enter the element to be enqueued: cde

Enter the priority: 6

|| bcd | 1 || --> || abc | 3 || --> || cde | 6 || --> NULL
```

```
MENU:
1. Enqueue
2. Dequeue
3. Display front and rear elements
4. Check if priority queue is empty
5. Size of the priority queue
6. Display priority queue
7. Exit
3

Front element is: || bcd | 1 ||
Rear element is: || cde | 6 ||
```

```
MENU:
1. Enqueue
2. Dequeue
3. Display front and rear elements
4. Check if priority queue is empty
5. Size of the priority queue
6. Display priority queue
7. Exit
4

Priority Queue is not empty.
```

```
MENU:
1. Enqueue
2. Dequeue
3. Display front and rear elements
4. Check if priority queue is empty
5. Size of the priority queue
6. Display priority queue
7. Exit
5

Size of the priority queue is: 3
```

```
MENU:
1. Enqueue
2. Dequeue
3. Display front and rear elements
4. Check if priority queue is empty
5. Size of the priority queue
6. Display priority queue
7. Exit
2

Dequeueing front element: || bcd | 1 ||

|| abc | 3 || --> || cde | 6 || --> NULL
```

```
MENU:
1. Enqueue
2. Dequeue
3. Display front and rear elements
4. Check if priority queue is empty
5. Size of the priority queue
6. Display priority queue
7. Exit
2

Dequeueing front element: || abc | 3 ||

|| cde | 6 || --> NULL
```

```
MENU:
1. Enqueue
2. Dequeue
3. Display front and rear elements
4. Check if priority queue is empty
5. Size of the priority queue
6. Display priority queue
7. Exit
2

Dequeueing front element: || cde | 6 ||

Priority Queue is empty! Nothing to display
```

```
MENU:
1. Enqueue
2. Dequeue
3. Display front and rear elements
4. Check if priority queue is empty
5. Size of the priority queue
6. Display priority queue
7. Exit
4

Priority Queue is empty!
```

FAIZAN CHOUDHARY

20BCS021

DSA LAB

7th December 2021

CODE: (code pasted in this format for readability)

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

struct list
{
    int info;
    struct list *next;
};
struct list *ptr, *temp, *p, *start=NULL, *rear=NULL;

void new_node (int n)
{
    ptr = (struct list *) malloc (sizeof(struct list));
    if (ptr==NULL)
    {
        cout<<"\nMemory could not be allocated!\n";
        return;
    }
    ptr->info = n;
    ptr->next = NULL;
}

int tot ()
{
    int c=0;
    if (start==NULL)
        return c;
    p=start;
    while (p != NULL)
    {
        p = p->next;
        c++;
    }
    return c;
}

void display ()
{
    if (tot()==0)
    {
        cout<<"\nList is empty, nothing to display!\n";
        return;
```

```
    }
    p=start->next;
    cout<<endl<<"List items: ";
    cout<<endl<<start->info;
    while (p!=NULL)
    {
        cout<<" -> "<<p->info;
        p=p->next;
    }
    cout<<" -> NULL"<<endl;
}

void insert_beg (int n)
{
    new_node(n);
    if (tot()==0)
    {
        start = ptr;
        rear = ptr;
    }
    else
    {
        // storing the previous first node's address to the next of the newly inserted
node
        temp = start;
        start = ptr;
        ptr->next = temp;
    }
    display();
}

void insert_end (int n)
{
    new_node(n);
    if (tot()==0)
    {
        start = ptr;
        rear = ptr;
        return ;
    }
    rear->next = ptr;
    rear = ptr;
    display();
}

void insert_pos (int n, int k)
{
    if (k==1)
    {
        insert_beg(n);
        return;
    }
    else if (k > tot())
    {
```

```
        insert_end(n);
        return;
    }

    new_node(n);
    int c = 1;
    if (tot()==0)
    {
        cout<<"\nList is empty, inserting at first position.\n";
        start = ptr;
        rear = ptr;
    }
    else
    {
        p = start;
        while (c < (k-1))
        {
            p=p->next;
            c++;
        }
        temp = p->next;
        p->next = ptr;
        ptr->next = temp;
    }
    display();
}

void del_beg ()
{
    if (tot()==0)
    {
        cout<<"\nList is empty, cannot delete!\n";
        return;
    }
    if (tot()==1)
    {
        start = NULL;
        rear = NULL;
        display();
        return ;
    }
    else
    {
        ptr = start;
        start = start->next;
        delete ptr;
    }
    display();
}

void del_end()
{
    if (tot()==0)
    {
```

```
        cout<<"\nList is empty, cannot delete!\n";
        return;
    }
    if (tot()==1)
    {
        start = NULL;
        rear = NULL;
        display();
        return ;
    }
    temp = start;
    while (temp->next != rear)
        temp = temp->next;
    ptr = rear;
    temp->next = NULL;
    rear = temp;
    delete ptr;

    display();
}

int search (int n)
{
    int pos=0;
    if (tot()==0)
        return -1;
    p=start;
    while (p != NULL)
    {
        pos++;
        if (p->info == n)
            return pos;
        p = p->next;
    }
    return 0;
}

int main()
{
    cout<<"\nFAIZAN CHOUDHARY\n20BCS021\n";

    int ch,n,k,key;
    while (true)
    {
        A:
        cout<<"\nMENU:\n1. Insert at beginning\n2. Insert at end\n3. Insert at given
position\n4. Deletion from beginning\n5. Deletion from end\n6. Total number of
elements\n7. Search item\n8. Display\n9. Exit\n";
        cin>>ch;
        switch (ch)
        {
            case 1: cout<<"\nEnter the element to be inserted: ";
                    cin>>n;
                    insert_beg(n);
```

```
                    break;
            case 2: cout<<"\nEnter the element to be inserted: ";
                    cin>>n;
                    insert_end(n);
                    break;
            case 3: cout<<"\nEnter the element to be inserted: ";
                    cin>>n;
                    cout<<"\nEnter the position to be inserted: ";
                    cin>>k;
                    insert_pos(n,k);
                    break;
            case 4: del_beg();
                    break;
            case 5: del_end();
                    break;
            case 6: cout<<"\nTotal number of elements in the list is: "<<tot()<<endl;
                    break;
            case 7: cout<<"\nEnter the key to be searched for: ";
                    cin>>key;
                    if (search(key) == -1)
                        cout<<"\nList is Empty!\n";
                    else if (search(key) == 0)
                        cout<<"\nElement not found in the list!\n";
                    else
                        cout<<"\nElement found in the list at position:
"<<search(key)<<endl;
                    break;
            case 8: display();
                    break;
            case 9: exit(0);
            default: cout<<"\nWrong choice! Enter again...\n";
                     goto A;
        }
    }
}
```

OUTPUT:

```
FAIZAN CHOUDHARY
20BCS021

MENU:
1. Insert at beginning
2. Insert at end
3. Insert at given position
4. Deletion from beginning
5. Deletion from end
6. Total number of elements
7. Search item
8. Display
9. Exit
1

Enter the element to be inserted: 44

List items:
44 -> NULL
```

```
MENU:
1. Insert at beginning
2. Insert at end
3. Insert at given position
4. Deletion from beginning
5. Deletion from end
6. Total number of elements
7. Search item
8. Display
9. Exit
2

Enter the element to be inserted: 55

List items:
44 -> 55 -> NULL
```

```
MENU:
1. Insert at beginning
2. Insert at end
3. Insert at given position
4. Deletion from beginning
5. Deletion from end
6. Total number of elements
7. Search item
8. Display
9. Exit
3

Enter the element to be inserted: 66

Enter the position to be inserted: 2

List items:
44 -> 66 -> 55 -> NULL
```

```
MENU:
1. Insert at beginning
2. Insert at end
3. Insert at given position
4. Deletion from beginning
5. Deletion from end
6. Total number of elements
7. Search item
8. Display
9. Exit
6

Total number of elements in the list is: 3
```

```
MENU:
1. Insert at beginning
2. Insert at end
3. Insert at given position
4. Deletion from beginning
5. Deletion from end
6. Total number of elements
7. Search item
8. Display
9. Exit
7

Enter the key to be searched for: 66

Element found in the list at position: 2
```

```
MENU:
1. Insert at beginning
2. Insert at end
3. Insert at given position
4. Deletion from beginning
5. Deletion from end
6. Total number of elements
7. Search item
8. Display
9. Exit
4

List items:
66 -> 55 -> NULL
```

```
MENU:
1. Insert at beginning
2. Insert at end
3. Insert at given position
4. Deletion from beginning
5. Deletion from end
6. Total number of elements
7. Search item
8. Display
9. Exit
5

List items:
66 -> NULL
```

```
MENU:
1. Insert at beginning
2. Insert at end
3. Insert at given position
4. Deletion from beginning
5. Deletion from end
6. Total number of elements
7. Search item
8. Display
9. Exit
5

List is empty, nothing to display!
```

FAIZAN CHOUDHARY

20BCS021

DSA LAB

14th December 2021

CODE: (code pasted in this format for readability)

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

struct list
{
    int info;
    struct list *next;
    struct list *prev;
};
struct list *ptr, *front=NULL, *rear=NULL, *p, *temp;

void create_node (int x)
{
    ptr=(struct list *) malloc (sizeof(struct list));
    if (ptr==NULL)
    {
        cout<<"\nMemory could not be allocated!\n";
        return;
    }
    ptr->info = x;
    ptr->next = NULL;
    ptr->prev = NULL;
}

int isEmpty ()
{
    if (front==NULL || rear==NULL)
        return 1;
    else
        return 0;
}

int size ()
{
    if (isEmpty()==1)
     return 0;
    else
    {
        int count=1;
        for (p=front; p!=rear; p=p->next)
         count++;
        return count;
```

```
    }
}

void display ()
{
    if (isEmpty()==1)
     cout<<"\nList is empty! Nothing to display\n";
    else
    {
        p=front;
        cout<<endl<<"NULL <- ";
        while (p->next != NULL)
        {
            cout<<p->info<<" <-> ";
            p=p->next;
        }
        cout<<rear->info;
        cout<<" -> NULL"<<endl;
    }
}

void display_rev ()
{
    if (isEmpty()==1)
     cout<<"\nList is empty! Nothing to display\n";
    else
    {
        p=rear;
        cout<<"NULL <- ";
        while (p->prev != NULL)
        {
            cout<<p->info<<" <-> ";
            p=p->prev;
        }
        cout<<front->info;
        cout<<" -> NULL"<<endl;
    }
}

void insert_beg (int n)
{
    create_node(n);
    if (front==NULL)
    {
        front=rear=ptr;
    }
    else
    {
        ptr->next = front;
        front->prev = ptr;
        ptr->prev = NULL;
        front = ptr;
    }
    display();
```

```
}

void insert_end (int n)
{
    create_node(n);
    if (front==NULL)
    {
        front=rear=ptr;
    }
    else
    {
        ptr->prev = rear;
        rear->next = ptr;
        ptr->next = NULL;
        rear = ptr;
    }
    display();
}

void insert_pos (int n, int k)
{
    if (k == 1)
    {
        insert_beg(n);
        return ;
    }
    else if (k > size())
    {
        insert_end(n);
        return ;
    }
    else
    {
        if (size()==0)
        {
            cout<<"\nList is empty, inserting at first position.\n";
            insert_beg(n);
            return ;
        }
        create_node(n);
        p = front;
        k--;
        while (k--)
            p = p->next;
        temp = p->prev;

        temp->next = ptr;
        ptr->prev = temp;
        p->prev = ptr;
        ptr->next = p;
    }
    display();
}
```

```
void del_beg ()
{
    if (isEmpty()==1)
    {
        cout<<"\nList is empty! Nothing to delete\n";
        return ;
    }
    ptr = front;
    front = front->next;
    if (front != NULL)
        front->prev = NULL;
    cout<<"\nDeleting element: "<<ptr->info<<endl;
    delete ptr;
    display();
}

void del_end ()
{
    if (isEmpty()==1)
    {
        cout<<"\nList is empty! Nothing to delete\n";
        return ;
    }
    ptr = rear;
    rear = ptr->prev;
    if (rear != NULL)
        rear->next = NULL;
    cout<<"\nDeleting element: "<<ptr->info<<endl;
    delete ptr;
    display();
}

void del_pos (int k) {
    int i=k;
    if (isEmpty()==1)
    {
        cout<<"\nList is empty! Nothing to delete\n";
        return ;
    }
    else if (k == 1) {
        del_beg();
        return ;
    }
    else if (k == size()) {
        del_end();
        return ;
    }
    ptr = front;
    i--;
    while (i-- && ptr != NULL)
        ptr = ptr->next;
    temp = ptr->prev;
    p = ptr->next;
    temp->next = p;
```

```
    p->prev = temp;
    cout<<"\nDeleting element: "<<ptr->info<<" at position "<<k<<endl;
    delete ptr;
    display();
}

void search (int key)
{
    if (isEmpty()==1)
    {
        cout<<"\nList empty!\n";
        return ;
    }
    bool flag = false;
    int k = 0;
    display();
    p = front;
    while (p != NULL) {
        k++;
        if (p->info == key) {
            flag = true;
            cout<<"\nElement found at position: "<<k<<" !\n";
        }
        p = p->next;
    }
    if (!flag)
        cout<<"\nElement not present in list!\n";
}

int main()
{
    cout<<"\nFAIZAN CHOUDHARY\n20BCS021\n";
    int ch,n,k;

    while (true)
    {
        A:
        cout<<"\nMENU:\n1. Insert at beginning\n2. Insert at end\n3. Insert at given
position\n4. Deletion from beginning\n5. Deletion from end\n6. Deletion from given
position\n7. Print list in reverse order\n8. Search element\n9. Display\n10. Exit\n";
        cin>>ch;
        switch (ch)
        {
        case 1: cout<<"\nEnter the element to be inserted: ";
                cin>>n;
                insert_beg(n);
                break;
        case 2: cout<<"\nEnter the element to be inserted: ";
                cin>>n;
                insert_end(n);
                break;
        case 3: cout<<"\nEnter the element to be inserted: ";
                cin>>n;
                C:
```

```
                cout<<"\nEnter position: ";
                cin>>k;
                if (k<=0)
                {
                    cout<<"\nEnter valid position!\n";
                    goto C;
                }
                insert_pos(n,k);
                break;
        case 4: del_beg();
                break;
        case 5: del_end();
                break;
        case 6: B:
                cout<<"\nEnter the position to be deleted: ";
                cin>>k;
                if (k<=0)
                {
                    cout<<"\nEnter valid position!\n";
                    goto B;
                }
                del_pos(k);
                break;
        case 7: cout<<"\nList elements: "<<endl;
                display();
                cout<<"\nList elements in reverse order: "<<endl;
                display_rev();
                break;
        case 8: cout<<"\nEnter element to be searched for: ";
                cin>>n;
                search (n);
                break;
        case 9: cout<<"\nList elements: "<<endl;
                display();
                break;
        case 10: exit(0);
        default: cout<<"\nWrong choice! Enter again...\n";
                 goto A;
        }
    }
}
```

OUTPUT:

```
FAIZAN CHOUDHARY
20BCS021

MENU:
1. Insert at beginning
2. Insert at end
3. Insert at given position
4. Deletion from beginning
5. Deletion from end
6. Deletion from given position
7. Print list in reverse order
8. Search element
9. Display
10. Exit
1

Enter the element to be inserted: 44

NULL <- 44 -> NULL
```

```
MENU:
1. Insert at beginning
2. Insert at end
3. Insert at given position
4. Deletion from beginning
5. Deletion from end
6. Deletion from given position
7. Print list in reverse order
8. Search element
9. Display
10. Exit
2

Enter the element to be inserted: 55

NULL <- 44 <-> 55 -> NULL
```

```
MENU:
1. Insert at beginning
2. Insert at end
3. Insert at given position
4. Deletion from beginning
5. Deletion from end
6. Deletion from given position
7. Print list in reverse order
8. Search element
9. Display
10. Exit
2

Enter the element to be inserted: 66

NULL <- 44 <-> 55 <-> 66 -> NULL
```

```
MENU:
1. Insert at beginning
2. Insert at end
3. Insert at given position
4. Deletion from beginning
5. Deletion from end
6. Deletion from given position
7. Print list in reverse order
8. Search element
9. Display
10. Exit
3

Enter the element to be inserted: 44

Enter position: 3

NULL <- 44 <-> 55 <-> 44 <-> 66 -> NULL
```

```
MENU:
1. Insert at beginning
2. Insert at end
3. Insert at given position
4. Deletion from beginning
5. Deletion from end
6. Deletion from given position
7. Print list in reverse order
8. Search element
9. Display
10. Exit
7

List elements:

NULL <- 44 <-> 55 <-> 44 <-> 66 -> NULL

List elements in reverse order:
NULL <- 66 <-> 44 <-> 55 <-> 44 -> NULL
```

```
MENU:
1. Insert at beginning
2. Insert at end
3. Insert at given position
4. Deletion from beginning
5. Deletion from end
6. Deletion from given position
7. Print list in reverse order
8. Search element
9. Display
10. Exit
8

Enter element to be searched for: 44

NULL <- 44 <-> 55 <-> 44 <-> 66 -> NULL

Element found at position: 1 !

Element found at position: 3 !
```

```
MENU:
1. Insert at beginning
2. Insert at end
3. Insert at given position
4. Deletion from beginning
5. Deletion from end
6. Deletion from given position
7. Print list in reverse order
8. Search element
9. Display
10. Exit
4

Deleting element: 44

NULL <- 55 <-> 44 <-> 66 -> NULL
```

```
MENU:
1. Insert at beginning
2. Insert at end
3. Insert at given position
4. Deletion from beginning
5. Deletion from end
6. Deletion from given position
7. Print list in reverse order
8. Search element
9. Display
10. Exit
5

Deleting element: 55

List is empty! Nothing to display
```

```
MENU:
1. Insert at beginning
2. Insert at end
3. Insert at given position
4. Deletion from beginning
5. Deletion from end
6. Deletion from given position
7. Print list in reverse order
8. Search element
9. Display
10. Exit
6

Enter the position to be deleted: 2

Deleting element: 44 at position 2

NULL <- 55 <-> 66 -> NULL
```

```
MENU:
1. Insert at beginning
2. Insert at end
3. Insert at given position
4. Deletion from beginning
5. Deletion from end
6. Deletion from given position
7. Print list in reverse order
8. Search element
9. Display
10. Exit
8

Enter element to be searched for: 5

List empty!
```

```
MENU:
1. Insert at beginning
2. Insert at end
3. Insert at given position
4. Deletion from beginning
5. Deletion from end
6. Deletion from given position
7. Print list in reverse order
8. Search element
9. Display
10. Exit
5

Deleting element: 66

NULL <- 55 -> NULL
```

```
MENU:
1. Insert at beginning
2. Insert at end
3. Insert at given position
4. Deletion from beginning
5. Deletion from end
6. Deletion from given position
7. Print list in reverse order
8. Search element
9. Display
10. Exit
10
```