Practical 02

Abhishek Gupta **2019450017**

1) Write a menu driven program to demonstrate Linear search.

Code :

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

class LSearch
{
    private:
    int noe,key,i;
    int arr[100];
    public:
    void GetData()
    {
        cout<<"Enter The Size of List(Maximum 100) : ";
        cin>>noe;
        for(i=0;i<noe;i++)
        {
            cout<<"Enter Data : ";
            cin>>arr[i];
        }
        cout<<endl<<"Enter Number to be Searched : ";
        cin>>key;
    }

    void add()
{
    noe+=1;
    arr[noe-1]=key;
}

    void ShowData()
```

Practical 02

Abhishek Gupta **2019450017**

```
    {
        for(int i=0;i<noe;i++)
        {
            cout<<arr[i]<<" ";
        }
    }

int Search()
{
    int flag=0;
    for(i=0;i<noe;i++)
    {
         if(key==arr[i])
            {
                flag=1;
                cout<<"The Number is Found At : "<<i<<"th Index
and "<<i+1<<"th position"<<endl;
                break;
            }
    }
    return flag;
}
};


int main()
{
    LSearch b=LSearch();
    b.GetData();
    if(b.Search() == 0)
    {
        b.add();
```

Practical 02

Abhishek Gupta **2019450017**

```
        b.ShowData();
    }
    return 0;
}
```

Output :

```
C:\Users\gupta\Desktop\new>1.exe
Enter The Size of List(Maximum 100) : 10
Enter Data : 25
Enter Data : 221
Enter Data : 32
Enter Data : 25
Enter Data : 321
Enter Data : 323
Enter Data : 3
Enter Data : 2
Enter Data : 22
Enter Data : 333

Enter Number to be Searched : 22
The Number is Found At : 8th Index and 9th position

C:\Users\gupta\Desktop\new>1.exe
Enter The Size of List(Maximum 100) : 5
Enter Data : 22
Enter Data : 32
Enter Data : 11
Enter Data : 22
Enter Data : 33

Enter Number to be Searched : 66
22      32      11      22      33      66
C:\Users\gupta\Desktop\new>
```

Practical 02

Abhishek Gupta **2019450017**

2) Write a menu driven program to demonstrate Binary search.

Code :

```
#include<iostream>
using namespace std;
class BSearch
{
    private:
    int noe,key,low,high,mid,i,j,temp;
    int arr[100];
    public:
    void GetData()
    {
        cout<<"Enter The Size of List(Maximum 100) : ";
        cin>>noe;
        for(i=0;i<noe;i++)
        {
            cout<<"Enter Data : ";
            cin>>arr[i];
        }
    }
      int Sorted()
    {
    for(int i=0;i<noe-1;i++)
        {
            if(arr[i]>arr[i+1])
                return 0;
        }
        return 1;
    }
```

Practical 02

Abhishek Gupta **2019450017**

```
void Sort()
{
    for(i=0;i<noe;i++)
    {
        if(Sorted())
        break;

        for(j=0;j<noe-1;j++)
        {
            if(arr[j]>arr[j+1])
            {
                temp=arr[j];
                arr[j]=arr[j+1];
                arr[j+1]=temp;
            }
        }
    }
}

 void ShowData()
{
    cout<<"Sorted Array :";
    for(int i=0;i<noe;i++)
    {
        cout<<arr[i]<<" ";
    }
    cout<<"\n\n";

    cout<<endl<<"Enter Number to be Searched : ";
    cin>>key;
}
```

Practical 02

Abhishek Gupta **2019450017**

```
    void Search()
    {
        int flag=0;
        low=0;
       high=noe-1;

   while(low <= high)
   {
       mid=(low+high)/2;

       cout<<mid;
       if(arr[mid] == key)
       {
           flag=1;
           cout<<"The Number is Found At : "<<mid<<"th Index and
"<<mid+1<<"th position"<<endl;
           break;
       }

       else if(arr[mid] > key)
       {
           high=mid-1;
       }

       else
       {
           low=mid+1;
       }

   }
   if (flag == 0)
   {
```

Practical 02

Abhishek Gupta **2019450017**

```
        cout<<"Number Not Present";
    }
}
};
int main()
{
    BSearch b=BSearch();
    b.GetData();
    b.Sort();
    b.ShowData();
    b.Search();
    return 0;
}
```

Output :

```
C:\Users\gupta\Desktop\new>b.exe
Enter The Size of List(Maximum 100) : 10
Enter Data : 11
Enter Data : 22
Enter Data : 3
Enter Data : 33
Enter Data : 44
Enter Data : 55
Enter Data : 66
Enter Data : 77
Enter Data : 88
Enter Data : 99
Sorted Array :3 11      22      33      44      55      66      77      88      99


Enter Number to be Searched : 88
478The Number is Found At : 8th Index and 9th position

C:\Users\gupta\Desktop\new>b.exe
Enter The Size of List(Maximum 100) : 5
Enter Data : 11
Enter Data : 22
Enter Data : 33
Enter Data : 44
Enter Data : 55
Sorted Array :11        22      33      44      55


Enter Number to be Searched : 66
234Number Not Present
```

Practical 02

Abhishek Gupta **2019450017**

3) Write a menu driven program to demonstrate Modulo division with linear probing.

Code :

```
#include<iostream>
using namespace std;
class Modulo
{
    public:
    void Hash(int *arr,int *key_arr,int n,int noe)
    {
        for(int i=0;i<n;i++)
        {
            if(i>=noe)
            {
                cout<<"Out of Bounds"<<endl;
                break;
            }
            int loc=key_arr[i]%noe;
            if(arr[loc] == 0)
            {
                arr[loc]=key_arr[i];
            }
            else
            {
                while(1)
                {
                    loc++;
                    if(loc >= noe)
                    {
                        loc=loc%noe;
                    }
```

Practical 02

Abhishek Gupta **2019450017**

```
                    if(arr[loc] == 0)
                    {
                        arr[loc]=key_arr[i];
                        break;
                    }
                }
            }

        }
    }

    void Display(int *arr,int noe)
    {
        for(int i=0;i<noe;i++)
        {
            cout<<i<<" "<<arr[i];
            cout<<endl;
        }

    }
};

int main()
{
    int n,noe;
    cout<<"Enter The Size of Array (Location Array) :";
    cin>>noe;
    int arr[noe] ={0};
    cout<<"Enter The Number of Elements to be Hashed :";
    cin>>n;
    int key_arr[n];
```

Practical 02

Abhishek Gupta **2019450017**

```
    for(int i=0;i<n;i++)
    {
        cout<<"Enter Data "<<i<<" : ";
        cin>>key_arr[i];
    }

    Modulo m=Modulo();
    m.Hash(arr,key_arr,n,noe);
    m.Display(arr,noe);
    return 0;
}
```

Output :

```
C:\Users\gupta\Desktop\new>m.exe
Enter The Size of Array (Location Array) :10
Enter The Number of Elements to be Hashed :10
Enter Data 0 : 88
Enter Data 1 : 99
Enter Data 2 : 65
Enter Data 3 : 332
Enter Data 4 : 21
Enter Data 5 : 255
Enter Data 6 : 3221
Enter Data 7 : 325
Enter Data 8 : 33
Enter Data 9 : 212
0 212
1 21
2 332
3 3221
4 33
5 65
6 255
7 325
8 88
9 99
```

Practical 02

Abhishek Gupta **2019450017**

4) Write a menu driven program to demonstrate Digit extraction with linear probing.

Code :

```
#include <iostream>
using namespace std;
class DigitExtraction
{
private:
    int arr[100], extract_arr[100], single_arr[100],
output_array[100];
    int choice, size, noe, n;

public:
    void get()
    {
        cout << "Enter the Size of Location Array : "
             << " ";
        cin >> size;

        for (int i = 0; i < size; i++)
        {
            arr[i] = -1;
        }

        cout << "Enter the Number of locations you want to
extract : "
             << " ";
        cin >> n;

        for (int i = 0; i < n; i++)
        {
            cout << "Enter Location : ";
```

Practical 02

Abhishek Gupta **2019450017**

```
            cin >> extract_arr[i];
        }

        cout << "How many numbers do you want to hash? : "
             << " ";
        cin >> noe;

        for (int i = 0; i < noe; i++)
        {
            cout << "Enter Element no " << i << " : ";
            cin >> choice;
            operations(choice);
        }
    }
    void display()
    {
        for (int i = 0; i < size; i++)
        {
            if (arr[i] == -1)
            {

            }
            else
            {
                cout << i<<" : "<<arr[i] << "\t";
            }
        }
        cout << endl;
    }

    void operations(int choice)
    {
```

Practical 02

Abhishek Gupta **2019450017**

```
        int location;
        int key = choice;
        location = count(key);
        cout << "Location is : "<<location << endl;
        if (arr[location] == -1)
        {
            arr[location] = choice;
        }
        else
        {
            while (1)
            {
                location++;
                if (arr[location] == -1)
                {
                    arr[location] = choice;
                    break;
                }
                if (location >= size)
                {
                    location = 0;
                }
            }
        }
    }
    int count(int key)
    {
        int temp, loc, value = 0, num, count = 0;
        temp = key;

        while (temp > 0)
        {
```

Practical 02

Abhishek Gupta **2019450017**

```
            temp = temp / 10;
            count++;
        }
        for (int i = 0; i < count; i++)
        {
            num = key % 10;
            single_arr[i] = num;
            key = key / 10;
        }

        for(int k=0;k<n;k++)
        {
            int m=extract_arr[k];
            output_array[k]=single_arr[m-1];
        }

        int start=0;
    int end=n-1;
    while (start < end)
    {
        int temp = output_array[start];
        output_array[start] = output_array[end];
        output_array[end] = temp;
        start++;
        end--;
    }
         for (int i = 0; i < n; i++)
         {
             value = value * 10 + output_array[i];
         }
         loc = value % size;
         return loc;
```

Practical 02

Abhishek Gupta **2019450017**

```
    }
};

int main()
{
    DigitExtraction d;
    d.get();
    d.display();
    return 0;
}
```

Output :

```
C:\Users\gupta\Desktop\new>d.exe
Enter the Size of Location Array :  100
Enter the Number of locations you want to extract :  3
Enter Location : 1
Enter Location : 2
Enter Location : 4
How many numbers do you want to hash? :  5
Enter Element no 0 : 12345
Location is : 45
Enter Element no 1 : 65412
Location is : 12
Enter Element no 2 :
32232
Location is : 32
Enter Element no 3 : 32563
Location is : 63
Enter Element no 4 : 32152
Location is : 52
12 : 65412      32 : 32232      45 : 12345      52 : 32152      63 : 32563

C:\Users\gupta\Desktop\new>
```

Practical 02

Abhishek Gupta **2019450017**

5) Write a menu driven program to demonstrate Fold boundary with linear probing .

//Manish,Shivam and I have created this program together

Code :

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

class Hash
{
    int arry[1000];
    int final[1000];
    int no_ele, size,divs, div_size;

public:
    Hash(int n, int s)
    {
        no_ele = n;
        size = s;

        for (int i = 0; i < s; i++)
            final[i] = -1;
    }

    void getdata()
    {
        cout << "**********************************\n";
        for (int i = 0; i < no_ele; i++)
        {
            cout << "Enter element : ";
            cin >> arry[i];
```

Practical 02

Abhishek Gupta **2019450017**

```
            if(arry[i]<0)
            {
                cout<<"\n----- Numbers cannot be negative
-----\n";
                cout<<"Enter a positive number : ";
                cin>>arry[i];
            }
        }
    }

    int linear_probing(int index)
    {
        while (final[index] != -1)
        {
            index++;
            if (index == size)
                index = 0;
        }
        return index;
    }

    //HASHING FUNCTION
    void fold_boundry()
    {
        cout << "**********************************\n";
        int index;
        for (int i = 0; i < no_ele; i++)
        {
            cout<<"\n--- Element "<<i+1<<" ---";
            if (i > size - 1)
            {
```

Practical 02

Abhishek Gupta **2019450017**

```
                cout << "\n--- " << arry[i] << " cannot be
stored as all locations are full ---\n";
            }
            else
            {
                index = divide(arry[i]);
                cout << "\nLocation = " << index << "\n";
                if (final[index] == -1)
                    final[index] = arry[i];
                else
                {
                    modulo_division(index, i);
                }
            }
        }
    }

    void modulo_division(int index, int i)
    {
        index = index % size;
        if (final[index] != -1)
            index = linear_probing(index);
        final[index] = arry[i];
        cout << "Location after Modulo-Division = " << index <<
"\n";
        cout << "****************\n";
    }

    //Functions for fold boundry
    int cnt(int n)
    {
        int temp = n, count = 0;
```

Practical 02

Abhishek Gupta **2019450017**

```
        while (temp > 0)
        {
            count++;
            temp /= 10;
        }
        return count;
    }

    int large(int a[])
    {
        int big = a[0];
        for (int i = 1; i < divs; i++)
        {
            if (big < a[i])
                big = a[i];
        }
        return big;
    }

    int rev(int n)
    {
        int r = 0;
        while (n > 0)
        {
            r = (r * 10) + (n % 10);
            n /= 10;
        }
    }

    int rev_arr(int n)
    {
        int r = 0;
```

Practical 02

Abhishek Gupta **2019450017**

```
        while (n > 0)
        {
            r = (r * 10) + (n % 10);
            n /= 10;
        }
        if (cnt(r) == 1)
            return r * 10;
        else
            return r;
    }

    int divide(int n)
    {
        int a_size = cnt(n), temp = n;
        div_size = cnt(size-1);
        int temp_size = a_size;
        int arr[a_size];
        int r;

        while (temp_size > 0)
        {
            arr[temp_size - 1] = temp % 10;
            temp /= 10;
            temp_size--;
        }

        cout << "\nDivided element : " << endl;
        for (int i = 0; i < a_size; i++)
            cout << arr[i] << "\t";

        divs =ceil((float) cnt(n) / cnt(size - 1));
```

Practical 02

Abhishek Gupta **2019450017**

```
        int diffrent[divs];
        temp_size = divs - 1;
        int count = 0;
        temp = 0;

        for (int i = a_size - 1; i >= 0; i--)
        {
            temp = arr[i] + temp * 10;
            diffrent[temp_size] = temp;
            count++;
            if (count == div_size)
            {
                temp_size--;
                count = 0;
                temp = 0;
            }
        }

        if(div_size>1)
        {
            for (int i = 0; i < divs; i++)
            {
                if (i != 0)
                {
                    if (i != divs - 1)
                        diffrent[i] = rev_arr(diffrent[i]);
                }
            }
        }

        if(diffrent[0]<10)
            diffrent[0]*=10;
```

Practical 02

Abhishek Gupta **2019450017**

```
    //cout<<"\n\n+++++++ Last element : "<<diffrent[divs]
    if(diffrent[divs-1]<10)
        diffrent[divs-1]*=10;

    cout << "\n\nDiffrentiated elements : \n";
    for (int i = 0; i < divs; i++)
        cout << diffrent[i] << "  ";

    cout << "\n****************";
    return without_carry(diffrent);
}

int without_carry(int arr[])
{
    int cn = cnt(large(arr)), add = 0, divisor = 1;

    for (int i = 0; i < divs; i++)
    {
        add = add + arr[i];
    }

    while (cn > 0)
    {
        divisor = divisor * 10;
        cn--;
    }

    cout<<"\nAdd with carry = "<<add;
    cout<<"\nAdd withiut carry ="<<add%divisor;
    return add % divisor;
}
```

Practical 02

Abhishek Gupta **2019450017**

```
    void showdata()
    {
        cout << "\nStored elements : \n";
        for (int i = 0; i < size; i++)
        {
            if (final[i] != -1)
                cout << "| " << i << " | " << final[i] << "
|\n";
        }
        cout << endl;
    }
};

int main()
{
    int no_ele, size;
    cout << "\n\nEnter No. of loactions : ";
    cin >> size;
    cout << "Enter no of elements : ";
    cin >> no_ele;

    Hash h(no_ele, size);
    h.getdata();
    h.fold_boundry();
    h.showdata();
}
```

Output :

Practical 02

Abhishek Gupta **2019450017**

```
Enter No. of loactions : 100
Enter no of elements : 5
********************************
Enter element : 12345
Enter element : 32145
Enter element : 21235
Enter element : 32223
Enter element : 25523
********************************

--- Element 1 ---
Divided element :
1       2       3       4       5

Diffrentiated elements :
10  23  54
****************
Add with carry = 87
Add withiut carry =87
Location = 87

--- Element 2 ---
Divided element :
3       2       1       4       5

Diffrentiated elements :
30  21  54
****************
Add with carry = 105
Add withiut carry =5
Location = 5

--- Element 3 ---
Divided element :
2       1       2       3       5

Diffrentiated elements :
20  12  53
****************
Add with carry = 85
Add withiut carry =85
Location = 85
```

Searching and Hashing

Practical 02

Abhishek Gupta **2019450017**

```
--- Element 4 ---
Divided element :
3       2       2       2       3

Diffrentiated elements :
30  22  32
****************
Add with carry = 84
Add withiut carry =84
Location = 84

--- Element 5 ---
Divided element :
2       5       5       2       3

Diffrentiated elements :
20  55  32
****************
Add with carry = 107
Add withiut carry =7
Location = 7

Stored elements :
| 5 | 32145 |
| 7 | 25523 |
| 84 | 32223 |
| 85 | 21235 |
| 87 | 12345 |


C:\Users\gupta\Desktop\new>
```

Practical 02

Abhishek Gupta **2019450017**

6) Write a menu driven program to demonstrate Mid square with linear probing.

Code :

```
#include<iostream>
using namespace std;
class Mid
{
    private:
    int arr[100];
    int choice,size,noe;
    public:
    void get()
    {
        cout<<"Enter the Size of Location Array : "<<" ";
        cin>>size;
        for(int i=0;i<size;i++)
        {
            arr[i]=-1;
        }
        cout<<"How many numbers do you want to hash? : "<<" ";
        cin>>noe;
        for(int i=0;i<noe;i++)
        {
            cout<<"Enter Element no "<<i<<" : ";
            cin>>choice;
            operations(choice);
        }
    }

    void display()
    {
```

Practical 02

Abhishek Gupta **2019450017**

```
        for (int i = 0; i < size; i++)
        {
            if (arr[i] == -1)
            {

            }
            else
            {
                cout << i<<" : "<<arr[i] << "\t";
            }
        }
        cout << endl;
    }

    void operations(int choice)
    {
        int key,location;
        key=choice*choice;
        location=count(key);
        cout<<location<<endl;

        if(arr[location] == -1)
        {
            arr[location]=choice;
        }
        else
        {
            while(1)
            {
                location++;
                if(arr[location]==-1)
                {
```

Practical 02

Abhishek Gupta **2019450017**

```
                    arr[location]=choice;
                    break;
                }
                if(location>=size)
                {
                    location=0;
                }
            }
        }
    }

    int count(int key)
    {
        int loc,temp,count=0;
        temp=key;
        while(temp>0)
        {
            temp=temp/10;
            count++;
        }
        if(count%2==0)
        {
            for(int i=1;i<((count+1)/2);i++)
            {
                key=key/10;
            }
            loc=key%100;
            loc=loc%size;
        }
        else if(count%2!=0)
        {
            for(int i=1;i<((count+1)/2);i++)
```

Practical 02

Abhishek Gupta **2019450017**

```
            {
                key=key/10;
            }
            loc=key%10;
        }
        return loc;
    }
};
int main()
{
    Mid m;
    m.get();
    m.display();
    return 0;
}
```

Output :

```
C:\Users\gupta\Desktop\new>m.exe
Enter the Size of Location Array :  100
How many numbers do you want to hash? :  5
Enter Element no 0 : 10
0
Enter Element no 1 : 11
2
Enter Element no 2 : 12
4
Enter Element no 3 : 13
6
Enter Element no 4 : 14
9
0 : 10  2 : 11  4 : 12  6 : 13  9 : 14

C:\Users\gupta\Desktop\new>
```