NAME-YASH GANDHI SE IT BATCH A 14

QUESTION-Implement the hash concept by hashing the given 'n' keys using modulo division method and solve the collision using

1- Chaining,

2-Linear Probing,

3-Quadratic probing

4- Double Hashing.

Show the insert, delete and search operations using each of these collision resolution technique and display the index if collision occurred and index at which collision resolved. Show the final key allocated table

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

int array[20];                    //GLOBALLY DECLARED ARRAY
void insert(int n,int x)          //FUNCTION TO INSERT KEY DEPENDING ON PROBING METHOD
{
 if(x==1)             //LINEAR
{
  int i=1,key,j=0;
  while(i)
  {
    printf("enter key\n");
    scanf("%d",&key);

    j=key%n;
    if(array[j]==-1||array[j]==0)
```

```
   array[j]=key;
   else
   {
    printf("collision for key %d occurs first at %d\n",key,j);   //COLLISION
    while(array[j]!=-1&&array[j]!=0)
    {
     j=(++j)%n;
    }
    array[j]=key;
    printf("collision for key %d is resolved at %d\n",key,j); //COLLISION RESOLVED
   }
   printf("enter 1 continue else 0\n");
   scanf("%d",&i);
  }
  return;
}
if(x==2)                    //QUADRATIC
{
   int i=1,key,j=0,q,p;
  while(i)
  {
   printf("enter key\n");
   scanf("%d",&key);

   j=key%n;
   if(array[j]==-1||array[j]==0)
   array[j]=key;
```

```
    else
    {
     q=j;
     p=1;
     printf("collision for key %d occurs first at %d\n",key,j);
     while(array[j]!=-1&&array[j]!=0)
     {
      j=(q+p+3*p*p)%n;
      p=p+1;
     }
     array[j]=key;
     printf("collision for key %d is resolved at %d\n",key,j);
    }
    printf("enter 1 continue else 0\n");
    scanf("%d",&i);
  }
  return;
}
if(x==3)                        //DOUBLE HASHING
{
 int i=1,key,j=0,q,y,p;
  while(i)
  {
    printf("enter key\n");
    scanf("%d",&key);
    j=key%n;
    if(array[j]==-1||array[j]==0)
```

```
   array[j]=key;
   else
   {
    q=j;
    p=1;
    y=8-key%8;
    printf("collision for key %d occurs first at %d\n",key,j);
    while(array[j]!=-1&&array[j]!=0)
    {
     j=(q+p*y)%n;
     p=p+1;
    }
    array[j]=key;
    printf("collision for key %d is resolved at %d\n",key,j);
   }
   printf("enter 1 continue else 0\n");
   scanf("%d",&i);
  }
  return;
 }
}

void display(int n)          //DISPLAY FUNCTION
{
 int i;
printf(“index\tkeys\n”);
 for(i=0;i<n;i++)
```

```
 {
  printf("%d\t %d\n",i,array[i]);
 }
return;
}

void search(int key,int n,int m)   //SEARCH FUNCTION
{
 if(m==1)                          //LINEAR
{
  int j,p,x=0,y=0;
  j=key%n;
  if(array[j]==key)
   {
   printf("the key is at %d\n",j);
   return;
   }
  else
  {
   p=j;
   printf(" during search, collision for key occurs first at %d\n",j);
   while(array[j]!=key)
   {
    j=(++j)%n;
    if(j==p)
    {
   x=1;
```

```
   break;
      }
   if(array[j]==0)
    {
     y=1;
     break;
    }
   }
    if(array[j]==key)
    printf("during search, collision for key resolved and found at %d\n",j);
    if(x==1)
    printf("SORRY probe over key not found\n");//ARRAY FULL
    if(y==1)
     printf("SORRY key not present --empty spaces present\n");//KEY ABSENT BUT SPACE PRESENT
   }
return;
}

if(m==2)              //QUADRATIC
{
    int j,p,x=0,y=0,q;
  j=key%n;
  if(array[j]==key)
   {
   printf("the key is at %d\n",j);
   return;
   }
```

```
else
{
  q=j;
   p=1;
 printf("during search, collision for key occurs first at %d\n",j);
 while(array[j]!=key)
 {
   j=(q+p+3*p*p)%n;
    p=p+1;
 if(j==q&&p>=n)
 {
x=1;
break;
    }
if(array[j]==0)
 {
   y=1;
   break;
 }
}

 if(array[j]==key)
 printf("during search, collision for key resolved and found at %d\n",j);
 if(x==1)
 printf("SORRY probe over key not found\n");
 if(y==1)
  printf("SORRY key not present --empty spaces present\n");
```

```
  }
return;
}
 if(m==3)              //DOUBLE HASHING
{
  int j,x,y=0,l,p,q;
  j=key%n;
  if(array[j]==key)
  {
  printf("the key is at %d\n",j);
  return;
  }
 else
 {
   q=j;
    p=1;
    l=8-key%8;
  printf("during search, collision for key occurs first at %d\n",j);
  while(array[j]!=key)
  {
     j=(q+p*l)%n;
     p=p+1;
   if(j==q&&p>=n)
   {
  x=1;
  break;
    }
```

```
  if(array[j]==0)
   {
     y=1;
     break;
   }
  }
   if(array[j]==key)
   {printf("during search, collision for key resolved and found at %d\n",j);
    return;}
   if(x==1)
   {printf("SORRY probe over key not found\n");
   return;}
   if(y==1)
   { printf("SORRY key not present --empty spaces present\n");
   return;}
  }
return;
 }
}

void del(int key,int n,int m)            //DELETE FUNCTION
{
  if(m==1)    //LINEAR
 {
  int j,p,x=0,y=0;
  j=key%n;
  if(array[j]==key)
```

```
 {
  array[j]=-1;
 return;
 }
else
{
 p=j;
 printf(" during delete, collision for key occurs first at %d\n",j);
 while(array[j]!=key)
 {
   j=(++j)%n;
   if(j==p)
   {
   x=1;
   break;
   }
  if(array[j]==0)
   {
    y=1;
    break;
   }
}
   if(array[j]==key)
   {
   printf(" during delete, collision for key resolved and found at %d\n",j);
   array[j]=-1;
    return;
```

```
    }
    if(x==1)
   {printf("SORRY probe over key not found\n");
    return;}
   if(y==1)
   {
    printf("SORRY key not present --empty spaces present\n");
     return;}
  }
return;
}

  if(m==2)              //QUADRATIC
{
   int j,p,x=0,y=0,q;
 j=key%n;
 if(array[j]==key)
  {
    array[j]=-1;
  return;
  }
 else
 {
   q=j;
    p=1;
  printf("during delete, collision for key occurs first at %d\n",j);
  while(array[j]!=key)
```

```
  {
    j=(q+p+3*p*p)%n;
     p=p+1;

   if(j==q&&p>=n)
   {
  x=1;
  break;
     }
  if(array[j]==0)
   {
     y=1;
     break;
   }
  }

   if(array[j]==key)
   {printf("during delete, collision for key resolved and found at %d\n",j);
     array[j]=-1;
     return;}
  if(x==1)
   printf("SORRY probe over key not found\n");
   if(y==1)
    printf("SORRY key not present --empty spaces present\n");
  }
return;
}
```

```
   if(m==3)      //DOUBLE HASHING
{
  int j,x,y=0,l,p,q;
  j=key%n;
  if(array[j]==key)
  {

  array[j]=-1;
  return;
  }
 else
 {
   q=j;
    p=1;
    l=8-key%8;

  printf("during delete, collision for key occurs first at %d\n",j);
  while(array[j]!=key)
  {
     j=(q+p*l)%n;
      p=p+1;

   if(j==q&&p>=n)
   {
  x=1;
  break;
```

```
      }
  if(array[j]==0)
   {
     y=1;
     break;
   }
  }

   if(array[j]==key)
   {printf("during delete, collision for key resolved and found at %d\n",j);
   array[j]=-1;
   return;}
   if(x==1)
   {printf("SORRY probe over key not found\n");
   return;}
   if(y==1)
   { printf("SORRY key not present --empty spaces present\n");
   return;}
  }
return;
 }
}

typedef struct node     //STRUCTURE FOR NODE DECLARED
{
 int k;
 struct node* next;
```

```
}node;

node* head[20];          //HEAD POINTER ARRAY DECLARED GLOBALLY
void cinsert(int n)    //INSERT FUNCTION FOR LINKLIST
{
int p=1,j=0,key;
 while(p)
{
  printf("enter key\n");
  scanf("%d",&key);
  j=key%n;
  node* newnode=(node*)malloc(sizeof(node*));
  newnode->k=key;
  newnode->next=NULL;
  if(head[j]==NULL)
    {
      head[j]=newnode;
    }
  else
 {
  printf("collision has occcured at %d\n",j);
  newnode->next=head[j];
  head[j]=newnode;
 }
   printf("enter 1 to continue 0 to exit\n");
  scanf("%d",&p);
}
```

```
return;
}

void cdisplay(int n)
{
   int i;
  printf("index\tkeys present\n");
  for(i=0;i<n;i++)
  {
    printf("%d\t",i);
    if(head[i]==NULL)
    {
      printf("0\n");
    }
    else
   {
    node* curr=head[i];
    while(curr->next!=NULL)
     {
       printf("%d-",curr->k);
       curr=curr->next;
     }
     printf("%d\n",curr->k);
    }
 }
return;
}
```

```
void csearch(int key,int n)      //SEARCH FUNCTION IN CHAINING RESOLUTION
{
  int j;
  j=key%n;
  int i=1;
  node* curr=head[j];
  while(curr->next!=NULL)
   {
    if(curr->k==key)
     {
       printf("key present at %d position on index %d\n",i,j);
        return;
      }
    curr=curr->next;
    i=i+1;
   }
 if(curr->k==key)
  {
     printf("key present at %d position on index %d\n",i,j);
        return;
  }
  else
  printf("key should be present at index %d but it is absent\n",j);
return;
}
```

```
void cdel(int key,int n)     //DELETE FUNCTION FOR LIST
{
    int j;
   j=key%n;
   int i=1;
   node* curr=head[j];
   if(head[j]->k==key)
   {
    printf("key at %d position on index %d and removed\n",i,j);
    head[j]=curr->next;
    free(curr);
    return;
   }
   node* prev=curr;
   while(curr->next!=NULL)
    {
     if(curr->k==key)
    {
       prev->next=curr->next;
         printf("key at %d position on index %d and removed\n",i,j);
      free(curr);
       return;
    }
     prev=curr;
     curr=curr->next;
     i=i+1;
    }
```

```
 if(curr->k==key)
  {
     printf("key present at %d position on index %d and removed\n",i,j);
     prev->next=NULL;
     free(curr);
      return;
  }
  else
  printf("key should be present at index %d but it is absent\n",j);
return;
}

void main()
{
 int i,n,p=1,b,key;
 for(i=0;i<20;i++)    //INITIALIZING ARRAYS
 {
 array[i]=0;
 head[i]=NULL;
 }
 int m;                    //RESOLUTION METHOD ASKED
 printf("enter 1 for chaining probe and 2 for open addressing probe\n");
 scanf("%d",&m);
if(m==1)                   //FOR CHAINING RESOLUTION METHOD
{
    printf("enter the length of head array\n");
                        scanf("%d",&n);
```

```
 while(p)
{
 printf("\nenter 0 to insert enter 1 for search 2 for delete 3 for display 4 to exit\n");
 scanf("%d",&b);
                    //ALL FUNCTIONS CALLED IN SWITCH CASE
 switch(b)
  {
   case 0:{
                      cinsert(n);
                      }break;
   case 1:{
                       printf("enter key to search \n");
                       scanf("%d",&key);
                       csearch(key,n);
                      }break;
  case 2:{
              printf("enter key to delete \n");
              scanf("%d",&key);
              cdel(key,n);
             }break;
  case 3:{
              cdisplay(n);
                 }break;

        case 4:  {
                    p=0;
                  }break;
```

```
                    default: printf("wrong");
                   }
                 }

}

if(m==2)          //OPEN ADDRESSING PROBE
{
 int x;
 printf("enter 1 for linear 2 for quadratic 3 for double hashing \n");
 scanf("%d",&x);    //SWITCH CASE FOR METHOD
 switch(x)
 {
  case 1: {
                  printf("enter array length\n");
                  scanf("%d",&n);
                   while(p)
                  {
                  printf("\nenter 0 to insert enter 1 for search 2 for delete 3 for display 4 to exit\n");
                  scanf("%d",&b);  //SWITCH CASE FOR FUNCTION CALLS
                   switch(b)
                   {
                    case 0:{
                                    insert(n,x);
                                    }break;
                    case 1:{
                                     printf("enter key to search \n");
```

```
                                scanf("%d",&key);
                                search(key,n,x);
                               }break;
                    case 2:{
                                printf("enter key to delete \n");
                                scanf("%d",&key);
                                del(key,n,x);
                               }break;
                    case 3:{
                                display(n);
                                   }break;

                          case 4:  {
                                      p=0;
                                     }break;
                          default: printf("wrong");
                         }
                    }
        }
        break;

case 2: {
                    printf("enter array length\n");
                    scanf("%d",&n);
                     while(p)
                    {
                    printf("\nenter 0 to insert enter 1 for search 2 for delete 3 for display 4 to exit\n");
```

```
scanf("%d",&b);


 switch(b)
 {
  case 0:{
                        insert(n,x);
                        }break;
  case 1:{
                         printf("enter key to search \n");
                         scanf("%d",&key);
                         search(key,n,x);
                        }break;
 case 2:{
               printf("enter key to delete \n");
               scanf("%d",&key);
               del(key,n,x);
              }break;
 case 3:{
               display(n);
                   }break;


        case 4:  {
                      p=0;
                     }break;
       default: printf("wrong");
     }
}
```

```
    }
    break;

case 3: {
                    printf("enter array length\n");
                    scanf("%d",&n);
                     while(p)
                    {
                    printf("\nenter 0 to insert enter 1 for search 2 for delete 3 for display 4 to exit\n");
                    scanf("%d",&b);

                     switch(b)
                      {
                       case 0:{
                                        insert(n,x);
                                        }break;
                       case 1:{
                                         printf("enter key to search \n");
                                         scanf("%d",&key);
                                         search(key,n,x);
                                        }break;
                      case 2:{
                                printf("enter key to delete \n");
                                scanf("%d",&key);
                                del(key,n,x);
                               }break;
                      case 3:{
```

```
                                display(n);
                                    }break;

                            case 4:  {
                                        p=0;
                                      }break;
                             default: printf("wrong");
                           }
                       }
          }
          break;
   default :printf("wrong");
   }
}
return;
 }
 /*
```

Output- chaining probe

```
enter 1 for chaining probe and 2 for open addressing probe
1
enter the length of head array
13

enter 0 to insert enter 1 for search 2 for delete 3 for display 4 to exit
0
enter key
18
```

enter 1 to continue 0 to exit

1

enter key

41

enter 1 to continue 0 to exit

1

enter key

22

enter 1 to continue 0 to exit

1

enter key

44

collision has occcured at 5

enter 1 to continue 0 to exit

1

enter key

59

enter 1 to continue 0 to exit

1

enter key

32

enter 1 to continue 0 to exit

1

enter key

31

collision has occcured at 5

enter 1 to continue 0 to exit

1

enter key

73

enter 1 to continue 0 to exit

0

enter 0 to insert enter 1 for search 2 for delete 3 for display 4 to exit

3

index	keys present
0	0
1	0
2	41
3	0
4	0
5	31-44-18
6	32
7	59
8	73
9	22
10	0
11	0
12	0

enter 0 to insert enter 1 for search 2 for delete 3 for display 4 to exit

1

enter key to search

45

key should be present at index 6 but it is absent

enter 0 to insert enter 1 for search 2 for delete 3 for display 4 to exit

1

enter key to search

44

key present at 2 position on index 5

enter 0 to insert enter 1 for search 2 for delete 3 for display 4 to exit

2

enter key to delete

44

key at 2 position on index 5 and removed

enter 0 to insert enter 1 for search 2 for delete 3 for display 4 to exit

3

index keys present

0 0

1 0

2 41

3 0

4 0

5 31-18

6 32

7 59

8 73

9 22

10 0

11 0

12 0

enter 0 to insert enter 1 for search 2 for delete 3 for display 4 to exit

0

enter key

62

enter 1 to continue 0 to exit

0

enter 0 to insert enter 1 for search 2 for delete 3 for display 4 to exit

0

enter key

42

enter 1 to continue 0 to exit

0

enter 0 to insert enter 1 for search 2 for delete 3 for display 4 to exit

3

index keys present

0 0

1 0

2 41

3 42

4 0

5 31-18

6 32

7 59

8 73

9 22

10 62

11 0

12 0

enter 0 to insert enter 1 for search 2 for delete 3 for display 4 to exit

0

enter key

49

collision has occcured at 10

enter 1 to continue 0 to exit

0

enter 0 to insert enter 1 for search 2 for delete 3 for display 4 to exit

3

index keys present

0 0

1 0

2 41

3 42

4 0

5 31-18

6 32

7 59

8 73

9 22

10 49-62

11 0

12 0

enter 0 to insert enter 1 for search 2 for delete 3 for display 4 to exit

4

OUTPUT LINEAR

enter 1 for chaining probe and 2 for open addressing probe

2

enter 1 for linear 2 for quadratic 3 for double hashing

1

enter array length

13

enter 0 to insert enter 1 for search 2 for delete 3 for display 4 to exit

0

enter key

18

enter 1 continue else 0

1

enter key

41

enter 1 continue else 0

1

enter key

22

enter 1 continue else 0

1

enter key

44

collision for key 44 occurs first at 5

collision for key 44 is resolved at 6

enter 1 continue else 0

1

enter key

59

enter 1 continue else 0

1

enter key

32

collision for key 32 occurs first at 6

collision for key 32 is resolved at 8

enter 1 continue else 0

2 1

enter key

31

collision for key 31 occurs first at 5

collision for key 31 is resolved at 10

enter 1 continue else 0

1

enter key

73

collision for key 73 occurs first at 8

collision for key 73 is resolved at 11

enter 1 continue else 0

1

enter key

12

enter 1 continue else 0

1

enter key

25

collision for key 25 occurs first at 12

collision for key 25 is resolved at 0

enter 1 continue else 0

0

enter 0 to insert enter 1 for search 2 for delete 3 for display 4 to exit

3

index keys

0 25

1 0

2 41

3 0

4 0

5 18

6 44

7 59

8 32

9 22

10 31

11 73

12 12

enter 0 to insert enter 1 for search 2 for delete 3 for display 4 to exit

1

enter key to search

100

during search, collision for key occurs first at 9

SORRY key not present --empty spaces present

enter 0 to insert enter 1 for search 2 for delete 3 for display 4 to exit

1

enter key to search

14

during search, collision for key occurs first at 1

SORRY key not present --empty spaces present

enter 0 to insert enter 1 for search 2 for delete 3 for display 4 to exit

2

enter key to delete

14

during delete, collision for key occurs first at 1

SORRY key not present --empty spaces present

enter 0 to insert enter 1 for search 2 for delete 3 for display 4 to exit

2

enter key to delete

32

during delete, collision for key occurs first at 6

during delete, collision for key resolved and found at 8

enter 0 to insert enter 1 for search 2 for delete 3 for display 4 to exit

3

index keys

0 25

1 0

2 41

3 0

4 0

5 18

6 44

7 59

8 -1

9 22

10 31

11 73

12 12

enter 0 to insert enter 1 for search 2 for delete 3 for display 4 to exit

0

enter key

21

enter 1 continue else 0

0

enter 0 to insert enter 1 for search 2 for delete 3 for display 4 to exit

3

index keys

0 25

1 0

2 41

3 0

4 0

5 18

6 44

7 59

8 21

9 22

10 31

11 73

12 12

enter 0 to insert enter 1 for search 2 for delete 3 for display 4 to exit

1

enter key to search

73

during search, collision for key occurs first at 8

during search, collision for key resolved and found at 11

enter 0 to insert enter 1 for search 2 for delete 3 for display 4 to exit

4

2)OUTPUT

enter 1 for chaining probe and 2 for open addressing probe

2

enter 1 for linear 2 for quadratic 3 for double hashing

1

enter array length

5

enter 0 to insert enter 1 for search 2 for delete 3 for display 4 to exit

0

enter key

11

enter 1 continue else 0

1

enter key

12

enter 1 continue else 0

1

enter key

23

enter 1 continue else 0

1

enter key

44

enter 1 continue else 0

1

enter key

55

enter 1 continue else 0

0

enter 0 to insert enter 1 for search 2 for delete 3 for display 4 to exit

3

index keys

0 55

1 11

2 12

3 23

4 44

enter 0 to insert enter 1 for search 2 for delete 3 for display 4 to exit

1

enter key to search

100

during search, collision for key occurs first at 0

SORRY probe over key not found

enter 0 to insert enter 1 for search 2 for delete 3 for display 4 to exit

1

enter key to search

23

the key is at 3

enter 0 to insert enter 1 for search 2 for delete 3 for display 4 to exit

4

OUTPUT-QUADRATIC 1

enter 1 for chaining probe and 2 for open addressing probe

2

enter 1 for linear 2 for quadratic 3 for double hashing

2

enter array length

11

enter 0 to insert enter 1 for search 2 for delete 3 for display 4 to exit

0

enter key

10

enter 1 continue else 0

1

enter key

22

enter 1 continue else 0

1

enter key

31

enter 1 continue else 0

1

enter key

4

enter 1 continue else 0

1

enter key

15

collision for key 15 occurs first at 4

collision for key 15 is resolved at 8

enter 1 continue else 0

1

enter key

28

enter 1 continue else 0

1

enter key

17

collision for key 17 occurs first at 6

collision for key 17 is resolved at 3

enter 1 continue else 0

1

enter key

88

collision for key 88 occurs first at 0

collision for key 88 is resolved at 2

enter 1 continue else 0

1

enter key

59

collision for key 59 occurs first at 4

collision for key 59 is resolved at 7

enter 1 continue else 0

0

enter 0 to insert enter 1 for search 2 for delete 3 for display 4 to exit

3

index keys

0 22

1 0

2 88

3 17

4 4

5 0

6 28

7 59

8 15

9 31

10 10

enter 0 to insert enter 1 for search 2 for delete 3 for display 4 to exit

1

enter key to search

17

during search, collision for key occurs first at 6

during search, collision for key resolved and found at 3

enter 0 to insert enter 1 for search 2 for delete 3 for display 4 to exit

1

enter key to search

88

during search, collision for key occurs first at 0

during search, collision for key resolved and found at 2

enter 0 to insert enter 1 for search 2 for delete 3 for display 4 to exit

2

enter key to delete

88

during delete, collision for key occurs first at 0

during delete, collision for key resolved and found at 2

enter 0 to insert enter 1 for search 2 for delete 3 for display 4 to exit

3

index keys

0 22

1 0

2 -1

3 17

4 4

5 0

6 28

7 59

8 15

9 31

10 10

enter 0 to insert enter 1 for search 2 for delete 3 for display 4 to exit

1

enter key to search

88

during search, collision for key occurs first at 0

SORRY key not present --empty spaces present

enter 0 to insert enter 1 for search 2 for delete 3 for display 4 to exit

0

enter key

99

collision for key 99 occurs first at 0

collision for key 99 is resolved at 2

enter 1 continue else 0

0

enter 0 to insert enter 1 for search 2 for delete 3 for display 4 to exit

3

index keys

0	22
1	0
2	99
3	17
4	4
5	0
6	28
7	59
8	15
9	31
10	10

enter 0 to insert enter 1 for search 2 for delete 3 for display 4 to exit

1

enter key to search

16

during search, collision for key occurs first at 5

SORRY key not present --empty spaces present

enter 0 to insert enter 1 for search 2 for delete 3 for display 4 to exit

4

2)output

enter 1 for chaining probe and 2 for open addressing probe

2

enter 1 for linear 2 for quadratic 3 for double hashing

2

enter array length

11

enter 0 to insert enter 1 for search 2 for delete 3 for display 4 to exit

0

enter key

12

enter 1 continue else 0

1

enter key

13

enter 1 continue else 0

1

enter key

14

enter 1 continue else 0

1

enter key

15

enter 1 continue else 0

1

enter key

16

enter 1 continue else 0

1

enter key

17

enter 1 continue else 0

1

enter key

18

enter 1 continue else 0

1

enter key

19

enter 1 continue else 0

1

enter key

40

collision for key 40 occurs first at 7

collision for key 40 is resolved at 0

enter 1 continue else 0

0

enter 0 to insert enter 1 for search 2 for delete 3 for display 4 to exit

3

index	key
0	40
1	12
2	13
3	14
4	15
5	16
6	17
7	18
8	19
9	0
10	0

enter 0 to insert enter 1 for search 2 for delete 3 for display 4 to exit

1

enter key to search

21

during search, collision for key occurs first at 10

SORRY key not present --empty spaces present

enter 0 to insert enter 1 for search 2 for delete 3 for display 4 to exit

1

enter key to search

20

during search, collision for key occurs first at 9

SORRY key not present --empty spaces present

enter 0 to insert enter 1 for search 2 for delete 3 for display 4 to exit

0

enter key

20

enter 1 continue else 0

1

enter key

21

enter 1 continue else 0

0

enter 0 to insert enter 1 for search 2 for delete 3 for display 4 to exit

3

index key

0 40

1 12

2 13

3 14

4 15

5 16

6 17

7 18

8 19

9 20

10 21

enter 0 to insert enter 1 for search 2 for delete 3 for display 4 to exit

1

enter key to search

23

during search, collision for key occurs first at 1

SORRY probe over key not found

enter 0 to insert enter 1 for search 2 for delete 3 for display 4 to exit

4

OUTPUT-DOUBLE HASHING

enter 1 for chaining probe and 2 for open addressing probe

2

enter 1 for linear 2 for quadratic 3 for double hashing

3

enter array length

13

enter 0 to insert enter 1 for search 2 for delete 3 for display 4 to exit

0

enter key

18

enter 1 continue else 0

1

enter key

41

enter 1 continue else 0

1

enter key

22

enter 1 continue else 0

1

enter key

44

collision for key 44 occurs first at 5

collision for key 44 is resolved at 0

enter 1 continue else 0

1

enter key

59

enter 1 continue else 0

1

enter key

32

enter 1 continue else 0

1

enter key

31

collision for key 31 occurs first at 5

collision for key 31 is resolved at 8

enter 1 continue else 0

1

enter key

73

collision for key 73 occurs first at 8

collision for key 73 is resolved at 3

enter 1 continue else 0

0

enter 0 to insert enter 1 for search 2 for delete 3 for display 4 to exit

3

index key

0 44

1 0

2 41

3 73

4 0

5 18

6 32

7 59

8 31

9 22

10 0

11 0

12 0

enter 0 to insert enter 1 for search 2 for delete 3 for display 4 to exit

1

enter key to search

31

during search, collision for key occurs first at 5

during search, collision for key resolved and found at 8

enter 0 to insert enter 1 for search 2 for delete 3 for display 4 to exit

1

enter key to search

100

during search, collision for key occurs first at 9

SORRY key not present --empty spaces present

enter 0 to insert enter 1 for search 2 for delete 3 for display 4 to exit

2

enter key to delete

22

enter 0 to insert enter 1 for search 2 for delete 3 for display 4 to exit

3

index key

0 44

1 0

2 41

3 73

4 0

5 18

6 32

7 59

8 31

9 -1

10 0

11 0

12 0

enter 0 to insert enter 1 for search 2 for delete 3 for display 4 to exit

0

enter key

100

enter 1 continue else 0

1

enter key

14

enter 1 continue else 0

0

enter 0 to insert enter 1 for search 2 for delete 3 for display 4 to exit

3

index key

0 44

1 14

2 41

3 73

4 0

5 18

6 32

7 59

8 31

9 100

10 0

11 0

12 0

enter 0 to insert enter 1 for search 2 for delete 3 for display 4 to exit

4

output 2

enter 1 for chaining probe and 2 for open addressing probe

2

enter 1 for linear 2 for quadratic 3 for double hashing

3

enter array length

13

enter 0 to insert enter 1 for search 2 for delete 3 for display 4 to exit

0

enter key

13

enter 1 continue else 0

1

enter key

14

enter 1 continue else 0

1

enter key

15

enter 1 continue else 0

1

enter key

16

enter 1 continue else 0

1

enter key

17

enter 1 continue else 0

1

enter key

18

enter 1 continue else 0

1

enter key

19

enter 1 continue else 0

1

enter key

20

enter 1 continue else 0

1

enter key

21

enter 1 continue else 0

1

enter key

22

enter 1 continue else 0

1

enter key

23

enter 1 continue else 0

1

enter key

24

enter 1 continue else 0

1

enter key

25

enter 1 continue else 0

0

enter 0 to insert enter 1 for search 2 for delete 3 for display 4 to exit

3

index key

0 13

1 14

2 15

3 16

4 17

5 18

6 19

7 20

8 21

9 22

10 23

11 24

12 25

enter 0 to insert enter 1 for search 2 for delete 3 for display 4 to exit

1

enter key to search

35

during search, collision for key occurs first at 9

SORRY probe over key not found

enter 0 to insert enter 1 for search 2 for delete 3 for display 4 to exit

1

enter key to search

25

the key is at 12

enter 0 to insert enter 1 for search 2 for delete 3 for display 4 to exit

2

enter key to delete

21

enter 0 to insert enter 1 for search 2 for delete 3 for display 4 to exit

0

enter key

12

collision for key 12 occurs first at 12

collision for key 12 is resolved at 8

enter 1 continue else 0

0

enter 0 to insert enter 1 for search 2 for delete 3 for display 4 to exit

3

index	key
0	13
1	14
2	15
3	16
4	17
5	18
6	19
7	20
8	12
9	22
10	23
11	24
12	25

enter 0 to insert enter 1 for search 2 for delete 3 for display 4 to exit

4

*/