

29. write a c program for hashing with linear probing?

**PROGRAM:**

```
#include<stdio.h>

#include<limits.h>

void insert(int ary[],int hFn, int size)
{
    int element,pos,n=0;
    printf("Enter key element to insert\n");
    scanf("%d",&element);
    pos = element%hFn;
    while(ary[pos]!= INT_MIN) {
        if(ary[pos]== INT_MAX)
            break;
        pos = (pos+1)%hFn;
        n++;
        if(n==size)
            break;
    }

    if(n==size)
        printf("Hash table was full of elements\nNo Place to insert this element\n\n");
    else
        ary[pos] = element; //Inserting element
    }

void search(int ary[],int hFn,int size){
    int element,pos,n=0;

    printf("Enter element you want to search\n");
    scanf("%d",&element);
```

```
pos = element%hFn;
```

```
while(n++ != size){
```

```
if(ary[pos]==element){
```

```
printf("Element found at index %d\n",pos);
```

```
break;
```

```
}
```

```
else
```

```
    if(ary[pos]==INT_MAX || ary[pos]!=INT_MIN)
```

```
        pos = (pos+1) %hFn;
```

```
}
```

```
if(--n==size) printf("Element not found in hash table\n");
```

```
}
```

```
void display(int ary[],int size){
```

```
int i;
```

```
printf("Index\tValue\n");
```

```
for(i=0;i<size;i++)
```

```
    printf("%d\t%d\n",i,ary[i]);
```

```
}
```

```
int main(){
```

```
int size,hFn,i,choice;
```

```
printf("Enter size of hash table\n");
```

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

```
int ary[size];

printf("Enter hash function [if mod 10 enter 10]\n");
scanf("%d",&hFn);

for(i=0;i<size;i++)
    ary[i]=INT_MIN; //Assigning INT_MIN indicates that cell is empty

do{
    printf("Enter your choice\n");
    printf(" 1-> Insert\n 2-> Display\n 3-> Searching\n 0-> Exit\n");
    scanf("%d",&choice);

    switch(choice){
    case 1:
        insert(ary,hFn,size);
        break;
    case 2:
        display(ary,size);
        break;
    case 3:
        search(ary,hFn,size);
        break;
    default:
        printf("Enter correct choice\n");
        break;
    }
}while(choice);
```

```

return 0;
}

```

## OUTPUT:

The screenshot shows a C++ IDE with the following code in `linearprobing.cpp`:

```

13 n++;
14 if(n==size)
15 break; // If table is full we should break, if not check this, loop will go
16 }
17
18 if(n==size)
19 printf("Hash table was full of elements\nNo Place to insert this element\n");
20 else
21 ary[pos] = element; //Inserting element
22 }
23
24 void search(int ary[],int hFn,int size){
25 int element,pos,n=0;
26
27 printf("Enter element you want to search\n");
28 scanf("%d",&element);
29
30 pos = element%hFn;
31
32 while(n++ != size){
33 if(ary[pos]==element){
34 printf("Element found at index %d\n",pos);
35 break;
36 }
37 }
38 }

```

The terminal output shows the following sequence of interactions:

```

Enter size of hash table
3
Enter hash function [if mod 10 enter 10]
10
Enter your choice
1-> Insert
2-> Display
3-> Searching
0-> Exit
1
Enter key element to insert
10
Enter your choice
1-> Insert
2-> Display
3-> Searching
0-> Exit
1
Enter key element to insert
1
Enter your choice
1-> Insert
2-> Display
3-> Searching
0-> Exit
1
Enter key element to insert
2
Enter your choice
1-> Insert

```

The screenshot shows the same C++ IDE with the same code as above. The terminal output continues from the previous state:

```

1
Enter key element to insert
2
Enter your choice
1-> Insert
2-> Display
3-> Searching
0-> Exit
1
Enter key element to insert
4
Hash table was full of elements
No Place to insert this element
Enter your choice
1-> Insert
2-> Display
3-> Searching
0-> Exit
2
Index Value
0 10
1 1
2 2
Enter your choice
1-> Insert
2-> Display
3-> Searching
0-> Exit

```

30.write a c program for bubble sorting?

**PROGRAM:**

```
#include <stdio.h>

int main()
{
    int a[15],i,j,n,temp;
    printf("\nEnter the size of array:") ;
    scanf("%d",&n);
    printf("\nEnter values for the array:\n");
    for(i=0;i<n;i++)
        scanf("%d",&a[i]);
    for(i=0;i<n-1;i++)
    {
        for(j=i+1;j<n;j++)
        {
            if(a[i]>a[j])
            {
                temp=a[i];
                a[i]=a[j];
                a[j]=temp;
            }
        }
    }
    printf("\nThe sorted array is:\n");
    for(i=0;i<n;i++)
        printf("%d ",a[i]);
    return 0;
}
```

## OUTPUT:

The image shows a screenshot of a C++ IDE (Dev-C++ 5.11) with a project named 'bubble.cpp'. The code implements a bubble sort algorithm. The output window shows the program's execution results.

```
3 {  
4     int a[15], i, j, n, temp;  
5     printf("\nENTER THE SIZE OF ARRAY:") ;  
6     scanf("%d", &n);  
7     printf("\nENTER VALUES FOR THE ARRAY:\n");  
8     for(i=0; i<n; i++)  
9         scanf("%d", &a[i]);  
10    for(i=0; i<n-1; i++)  
11    {  
12        for(j=i+1; j<n; j++)  
13        {  
14            if(a[i]>a[j])  
15            {  
16                temp=a[i];  
17                a[i]=a[j];  
18                a[j]=temp;  
19            }  
20        }  
21    }  
22    printf("\nTHE SORTED ARRAY IS:\n");  
23    for(i=0; i<n; i++)  
24        printf("%d ", a[i]);  
25    return 0;  
26 }
```

Compilation results...

- Errors: 0
- Warnings: 0
- Output Filename: C:\Users\reddy\OneDrive\Documents\bubble.exe
- Output Size: 128.7705078125 KiB
- Compilation Time: 0.22s

Output:

```
ENTER THE SIZE OF ARRAY:5  
ENTER VALUES FOR THE ARRAY:  
3 2 4 1 0  
  
THE SORTED ARRAY IS:  
0 1 2 3 4  
-----  
Process exited after 8.624 seconds with return value 0  
Press any key to continue . . .
```

31.write the c program for selection sorting:

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

```
int main()
```

```
{
```

```
    int array[10];
```

```
    int i, j, N, temp;
```

```
    int findmax(int b[10], int k);
```

```
    void exchang(int b[10], int k);
```

```
    printf("Enter the value of N\n");
```

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

```
    printf("Enter the elements one by one\n");
```

```
    for(i=0; i<N ; i++)
```

```
    {
```

```
        scanf("%d",&array[i]);
```

```
    }
```

```
    printf("Input array elements\n");
```

```
    for(i=0; i<N ; i++)
```

```
    {
```

```
        printf("%d\n",array[i]);
```

```
    }
```

```
    exchang(array,N);
```

```
    printf("Sorted array is...\n");
```

```

    for(i=0; i< N ; i++)
    {
        printf("%d\n",array[i]);
    }
    return 0;
}

```

```

int findmax(int b[10], int k)
{
    int max=0,j;
    for(j = 1; j <= k; j++)
    {
        if ( b[j] > b[max])
        {
            max = j;
        }
    }
    return(max);
}

```

```

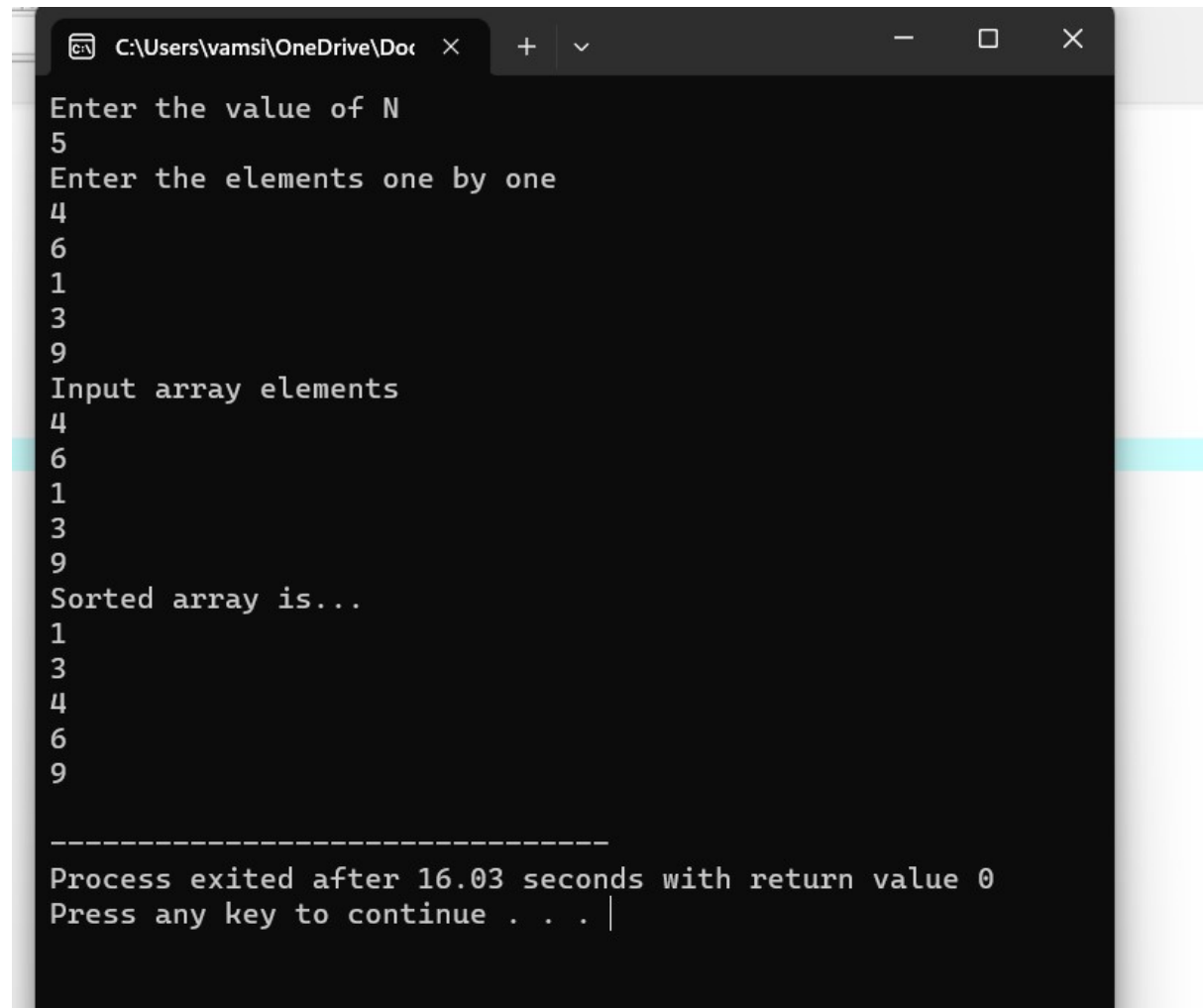
void exchang(int b[10],int k)
{
    int temp, big, j;
    for ( j=k-1; j>=1; j--)
    {
        big = findmax(b,j);
        temp = b[big];
        b[big] = b[j];
        b[j] = temp;
    }
    return;
}

```



}

### Output:



```
C:\Users\vamsi\OneDrive\Doc × + ▾  
Enter the value of N  
5  
Enter the elements one by one  
4  
6  
1  
3  
9  
Input array elements  
4  
6  
1  
3  
9  
Sorted array is...  
1  
3  
4  
6  
9  
-----  
Process exited after 16.03 seconds with return value 0  
Press any key to continue . . . |
```

**32.**write a c program for insertion sorting:

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

```
int main()
```

```
{
```

```
    int A[20],n,Temp,i,j;
```

```
    printf("\n\t\t\t\t-----INSERTION SORT-----\n\n");
```

```
    printf("\n\n ENTER THE NUMBER OF TERMS...: ");
```

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

```
    printf("\n ENTER THE ELEMENTS OF THE ARRAY...\n");
```

```
    for(i=0; i < n;i++)
```

```
    {
```

```
        scanf("%d", &A[i]);
```

```
    }
```

```
    for(i=1; i< n; i++)
```

```
    {
```

```
        Temp = A[i];
```

```
        j = i-1;
```

```
        while(Temp < A[j] && j>=0)
```

```
        {
```

```
            A[j+1] = A[j];
```

```
            j = j-1;
```

```
        }
```

```
        A[j+1] = Temp;
```

```
    }
```

```
    printf("\n\t\t\t\t-----INSERTION SORTED ELEMENTS-----\n\n");
```

```
    printf("\nTHE ASCENDING ORDER LIST IS...:");
```

```
    for(i=0; i < n; i++)
```

```
    {
```

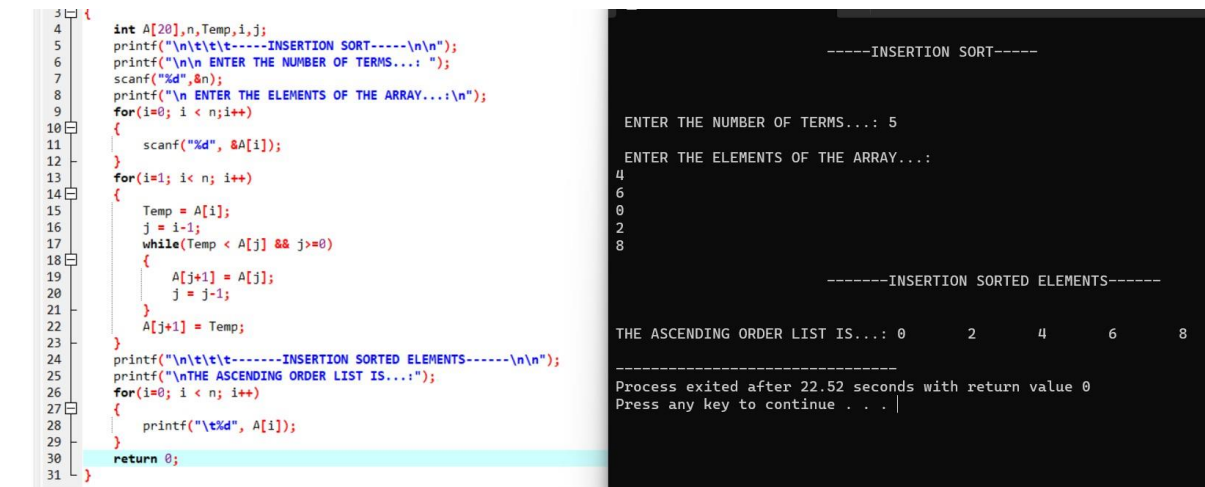
```
        printf("\t%d", A[i]);
```

```
    }
```

```
    return 0;
```

}

Output:



The image shows a C program for Insertion Sort on the left and its output on the right. The program defines an array A of size 20, prompts the user for the number of terms (n) and the elements of the array, and then performs an insertion sort. The output shows the sorted array elements: 0, 2, 4, 6, 8.

```
31 {  
4 int A[20], n, Temp, i, j;  
5 printf("\n\t\t\t\t\t-----INSERTION SORT-----\n\n");  
6 printf("\n\n ENTER THE NUMBER OF TERMS...: ");  
7 scanf("%d", &n);  
8 printf("\n ENTER THE ELEMENTS OF THE ARRAY...\n");  
9 for(i=0; i < n; i++)  
10 {  
11     scanf("%d", &A[i]);  
12 }  
13 for(i=1; i < n; i++)  
14 {  
15     Temp = A[i];  
16     j = i-1;  
17     while(Temp < A[j] && j>=0)  
18     {  
19         A[j+1] = A[j];  
20         j = j-1;  
21     }  
22     A[j+1] = Temp;  
23 }  
24 printf("\n\t\t\t\t\t-----INSERTION SORTED ELEMENTS-----\n\n");  
25 printf("\nTHE ASCENDING ORDER LIST IS...:");  
26 for(i=0; i < n; i++)  
27 {  
28     printf("\t%d", A[i]);  
29 }  
30 return 0;  
31 }
```

-----INSERTION SORT-----  
  
ENTER THE NUMBER OF TERMS...: 5  
ENTER THE ELEMENTS OF THE ARRAY...:  
4  
6  
0  
2  
8  
  
-----INSERTION SORTED ELEMENTS-----  
  
THE ASCENDING ORDER LIST IS...: 0        2        4        6        8  
  
-----  
Process exited after 22.52 seconds with return value 0  
Press any key to continue . . . |

33.write a c program for merge sorting:

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

```
#include<conio.h>
```

```
int arr[20];
```

```
void main()
```

```
{
```

```
int n,i;
```

```
clrscr();
```

```
printf("\n\t\t\t\t\t-----Merge Sorting-----\n\n");
```

```
printf("Enter the size of array\n");
```

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

```
printf("Enter the elements:\n");
```

```
for(i=0; i < n; i++)
```

```
{
```

```
scanf("%d",&arr[i]);
```

```
}
```

```
merge_sort(arr,0,n-1);
```

```
printf("\n\n\t\t\t\t\t-----Merge Sorted Elements-----\n\n");
```

```
printf("Sorted array:\n");
```

```

    for(i=0; i < n; i++)
    {
        printf("\t%d",arr[i]);
    }
    getch();
}

int merge_sort(int arr[],int low,int high)
{
    int mid;
    if(low < high)
    {
        mid=(low+high)/2;
        merge_sort(arr,low,mid);
        merge_sort(arr,mid+1,high);
        merge(arr,low,mid,high);
    }
}

int merge(int arr[],int l,int m,int h)
{
    int arr1[10],arr2[10];
    int n1,n2,i,j,k;
    n1=m-l+1;
    n2=h-m;
    for(i=0; i < n1; i++)
    {
        arr1[i]=arr[l+i];
    }
    for(j=0; j < n2; j++)
    {
        arr2[j]=arr[m+j+1];
    }
}

```

```

    }
    arr1[i]=9999;
    arr2[j]=9999;
    i=0;
    j=0;
    for(k=1; k <=h; k++)
    {
        if(arr1[i]<=arr2[j])
            arr[k]=arr1[i++];
        else
            arr[k]=arr2[j++];
    }
}

```

Output:

```

-----Counting Sort-----

Enter the number of input : 6
Enter the elements to be sorted :
2
4
3
0
2
1

----Sorted Array Using Counting Sort----

The Sorted array is :  0      1      2      2      3      4_

```

34.write a c program for quick sorting:

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

```
int quicksort(int arr[], int lb, int ub);
```

```
int main()
```

```
{
```

```
    int arr[20], n, i;
```

```
    printf("\n\t\t\t-----Quick Sort-----\n\n");
```

```
    printf("Enter the size of the array:");
```

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

```
    printf("Enter the elements to be sorted:\n");
```

```
    for(i=0;i < n;i++)
```

```
        scanf("%d",&arr[i]);
```

```
    quicksort(arr, 0, n-1);
```

```
    printf("\n\t\t\t-----Quick Sorted Elements-----\n\n");
```

```
    printf("Sorted array:");
```

```
for(i = 0; i < n; i++)
```

```
printf("\t%d ",arr[i]);
```

```
return 0;
```

```
}
```

```
int quicksort(int arr[], int lb, int ub)
```

```
{
```

```
    int pivot, i, j, temp;
```

```
    if(lb < ub)
```

```
    {
```

```
        pivot = lb;
```

```
        i = lb;
```

```
        j = ub;
```

```
        while(i < j)
```

```
        {
```

```
            while(arr[i] <= arr[pivot] && i <= ub)
```

```
                i++;
```

```
        while(arr[j] > arr[pivot] && j >= lb)

            j--;

        if(i < j)

        {

            temp = arr[i];

            arr[i] = arr[j];

            arr[j] = temp;

        }

    }

    temp = arr[j];

    arr[j] = arr[pivot];

    arr[pivot] = temp;

    quicksort(arr, lb, j-1);

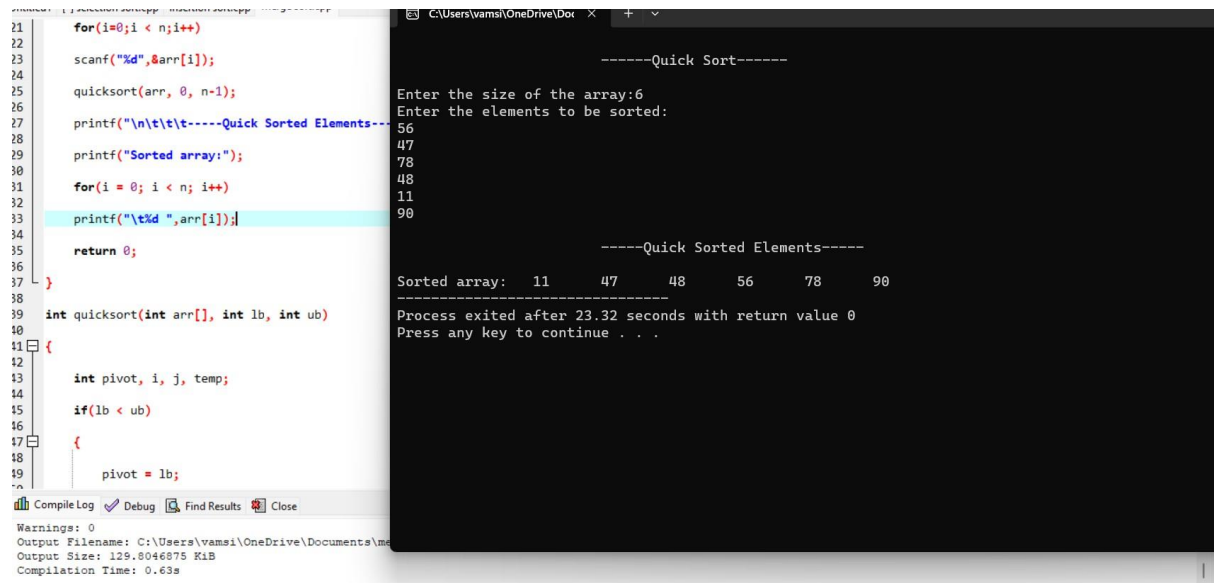
    quicksort(arr, j+1, ub);

}
```



}

Out put:



The image shows a C++ IDE with a source code editor on the left and a console window on the right. The source code implements a quicksort algorithm. The console output shows the program's execution, including prompts for array size and elements, the sorted array, and a message indicating the process exited after 23.32 seconds.

```
21     for(i=0; i < n; i++)
22     {
23         scanf("%d", &arr[i]);
24     }
25     quicksort(arr, 0, n-1);
26
27     printf("\n\t\t\t\t\t----Quick Sorted Elements----\n");
28     printf("Sorted array:");
29
30     for(i = 0; i < n; i++)
31     {
32         printf("\t%d ", arr[i]);
33     }
34
35     return 0;
36 }
37
38
39 int quicksort(int arr[], int lb, int ub)
40 {
41     if(lb < ub)
42     {
43         int pivot, i, j, temp;
44
45         if(lb < ub)
46         {
47             pivot = lb;
48             i = lb + 1;
49             j = ub;
50             while(i <= j)
51             {
52                 while(arr[i] <= arr[pivot])
53                     i++;
54                 while(arr[j] > arr[pivot])
55                     j--;
56                 if(i < j)
57                 {
58                     temp = arr[i];
59                     arr[i] = arr[j];
60                     arr[j] = temp;
61                 }
62             }
63             temp = arr[pivot];
64             arr[pivot] = arr[j];
65             arr[j] = temp;
66             quicksort(arr, lb, i-1);
67             quicksort(arr, j+1, ub);
68         }
69     }
70 }
```

-----Quick Sort-----  
Enter the size of the array:6  
Enter the elements to be sorted:  
56  
47  
78  
48  
11  
90  
-----Quick Sorted Elements-----  
Sorted array: 11 47 48 56 78 90  
-----  
Process exited after 23.32 seconds with return value 0  
Press any key to continue . . .

Compile Log | Debug | Find Results | Close  
Warnings: 0  
Output Filename: C:\Users\vamsi\OneDrive\Documents\me  
Output Size: 129.8046875 KiB  
Compilation Time: 0.63s