1. find the position of a target value within a sorted array using Binary search.

#include<stdio.h>

int binarySearch(int[], int, int, int);

void main ()

{

int arr[10] = {16, 19, 20, 23, 45, 56, 78, 90, 96, 100};

int item, location=-1;

printf("Enter the item which you want to search ");

scanf("%d",&item);

location = binarySearch(arr, 0, 9, item);

if(location != -1)

{

printf("Item found at location %d",location);

}

else

{

printf("Item not found");

}

}

int binarySearch(int a[], int beg, int end, int item)

{

int mid;

if(end >= beg)

{

mid = (beg + end)/2;

if(a[mid] == item)

{

return mid+1;

}

else if(a[mid] < item)

{

return binarySearch(a,mid+1,end,item);

}

else

{

return binarySearch(a,beg,mid-1,item);

}

}

return -1;

}

int binarySearch(int a[], int beg, int end, int item)

{

int mid;

if(end >= beg)

{

mid = (beg + end)/2;

if(a[mid] == item)

{

return mid+1;

}

else if(a[mid] < item)

{

return binarySearch(a,mid+1,end,item);

}

else

{

return binarySearch(a,beg,mid-1,item);

}

}

return -1;

}

2. find the position of a target value within a array using Linear search.

#include<stdio.h>

void main ()

{

int a[10] = {10, 23, 40, 1, 2, 0, 14, 13, 50, 9};

int item, i,flag;

printf("\nEnter Item which is to be searched\n");

scanf("%d",&item);

for (i = 0; i< 10; i++)

{

if(a[i] == item)

{

flag = i+1;

break;

}

else

flag = 0;

}

if(flag != 0)

{

printf("\nItem found at location %d\n",flag);

}

else

{

printf("\nItem not found\n");

}

}

output:

Enter Item which is to be searched

22

Item not found

3. sort a list of elements using the bubble sort algorithm (Ascending and Descending).

#include <stdio.h>

#define MAX 100

int main()

{

int arr[MAX],limit;

int i,j,temp;

printf("Enter total number of elements: ");

scanf("%d",&limit);

printf("Enter array elements: \n");

for(i=0; i<limit; i++)

{

printf("Enter element %3d: ",i+1);

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

}

for(i=0; i<(limit-1); i++)

{

for(j=0; j<(limit-i-1); j++)

{

if(arr[j]>arr[j+1])

{

temp=arr[j];

arr[j]=arr[j+1];

arr[j+1]=temp;

}

}

}

printf("Array elements in Ascending Order:\n");

for(i=0; i<limit; i++)

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

printf("\n");

for(i=0; i<(limit-1); i++)

{

for(j=0; j<(limit-i-1); j++)

{

if(arr[j]<arr[j+1])

{

temp=arr[j];

arr[j]=arr[j+1];

arr[j+1]=temp;

}

}

}

printf("Array elements in Descending Order:\n");

for(i=0; i<limit; i++)

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

printf("\n");

return 0;

}

OUTPUT:

Enter total number of elements: 10

Enter array elements:

Enter element 1: 12

Enter element 2: 34

Enter element 3: 43

Enter element 4: 32

Enter element 5: 21

Enter element 6: 1

Enter element 7: 11

Enter element 8: 2

Enter element 9: 3

Enter element10: 100

Array elements in Ascending Order:

1 2 3 11 12 21 32 34 43 100

Array elements in Descending Order:

100 43 34 32 21 12 11 3 2 1

4. sort a list of elements using the selection sort algorithm (Ascending and Descending)

#include<stdio.h>

int smallest(int[],int,int);

void main ()

{

int a[10] = {10, 9, 7, 101, 23, 44, 12, 78, 34, 23};

int i,j,k,pos,temp;

for(i=0;i<10;i++)

{

pos = smallest(a,10,i);

temp = a[i];

a[i]=a[pos];

a[pos] = temp;

}

printf("\nprinting sorted elements...\n");

for(i=0;i<10;i++)

{

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

}

}

int smallest(int a[], int n, int i)

{

int small,pos,j;

small = a[i];

pos = i;

for(j=i+1;j<10;j++)

{

if(a[j]<small)

{

small = a[j];

pos=j;

}

}

return pos;

}

OUTPUT:

printing sorted elements...

7

9

10

12

23

23

34

44

78

101

6. sort a list of elements using the radix sort algorithm (Ascending and

Descending)

#include <stdio.h>

int largest(int a[]);

void radix\_sort(int a[]);

void main()

{

int i;

int a[10]={90,23,101,45,65,23,67,89,34,23};

radix\_sort(a);

printf("\n The sorted array is: \n");

for(i=0;i<10;i++)

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

}

int largest(int a[])

{

int larger=a[0], i;

for(i=1;i<10;i++)

{

if(a[i]>larger)

larger = a[i];

}

return larger;

}

void radix\_sort(int a[])

{

int bucket[10][10], bucket\_count[10];

int i, j, k, remainder, NOP=0, divisor=1, larger, pass;

larger = largest(a);

while(larger>0)

{

NOP++;

larger/=10;

}

for(pass=0;pass<NOP;pass++) // Initialize the buckets

{

for(i=0;i<10;i++)

bucket\_count[i]=0;

for(i=0;i<10;i++)

{

// sort the numbers according to the digit at passth place

remainder = (a[i]/divisor)%10;

bucket[remainder][bucket\_count[remainder]] = a[i];

bucket\_count[remainder] += 1;

}

// collect the numbers after PASS pass

i=0;

for(k=0;k<10;k++)

{

for(j=0;j<bucket\_count[k];j++)

{

a[i] = bucket[k][j];

i++;

}

}

divisor \*= 10;

}

}

OUTPUT:

The sorted array is:

23 23 23 34 45 65 67 89 90 101