OBJECTIVE: Write a program to implement linear search algorithm in C.

PROGRAM:

#include <stdio.h>

int linear\_search(int arr[], int size, int key);

int main() {

int arr[7] = {4, 7, 1, 3, 9, 5, 2};

int key;

printf("Enter the element to search: ");

scanf("%d", &key);

int result = linear\_search(arr, 7, key);

result != -1 ? printf("Element found at index %d\n", result) : printf("Element not found\n");

return 0;

}

int linear\_search(int arr[], int size, int key) {

for(int i = 0; i < size; i++) {

if(arr[i] == key) {

return i;

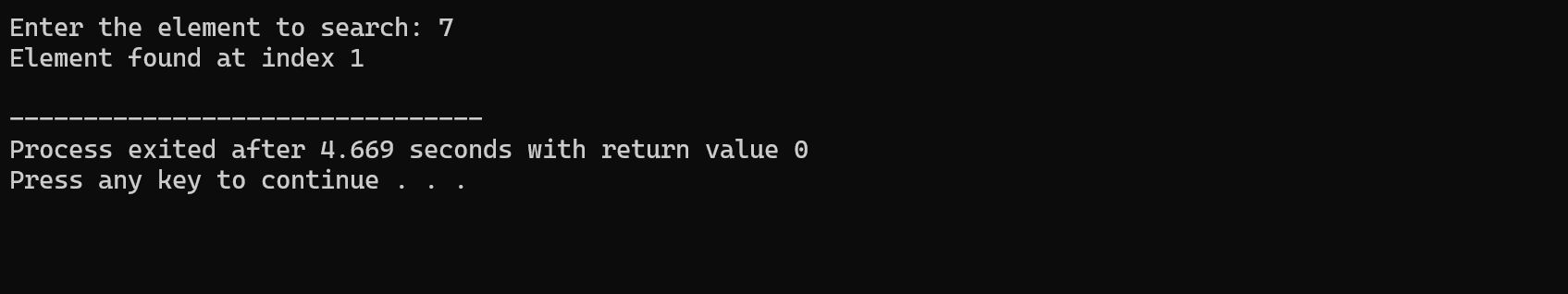
}

}

return -1;

}

OUTPUT:



OBJECTIVE: Write a program to implement bubble sort without flag.

PROGRAM:

#include<stdio.h>

#include <stdlib.h>

#define NUM\_NUMBERS 5000

void interchange(int a[],int i,int j)

{

int p=a[i];

a[i]=a[j];

a[j]=p;

}

void bubbleSort(int arr[], int n) {

int i,j;

for (i = 0; i < NUM\_NUMBERS-1; i++) {

for (j = 0; j < NUM\_NUMBERS- i-1 ; j++) {

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

interchange(arr, j, j + 1);

}

}

}

int main()

{

int i;

int \*arr=(int \*)malloc(NUM\_NUMBERS \*sizeof(int));

const char \*file\_path="D:\\7th semester\\random\_numbers.txt";

FILE \*file=fopen(file\_path,"r");

for(i=0;i<NUM\_NUMBERS;i++)

{

if(fscanf(file,"%d",&arr[i])!=1)

{

printf("Error\n");

fclose(file);

}

}

for(i=NUM\_NUMBERS-50;i<NUM\_NUMBERS;i++)

{

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

}

printf("\n\n");

bubbleSort(arr,NUM\_NUMBERS);

for(i=NUM\_NUMBERS-50;i<NUM\_NUMBERS;i++)

{

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

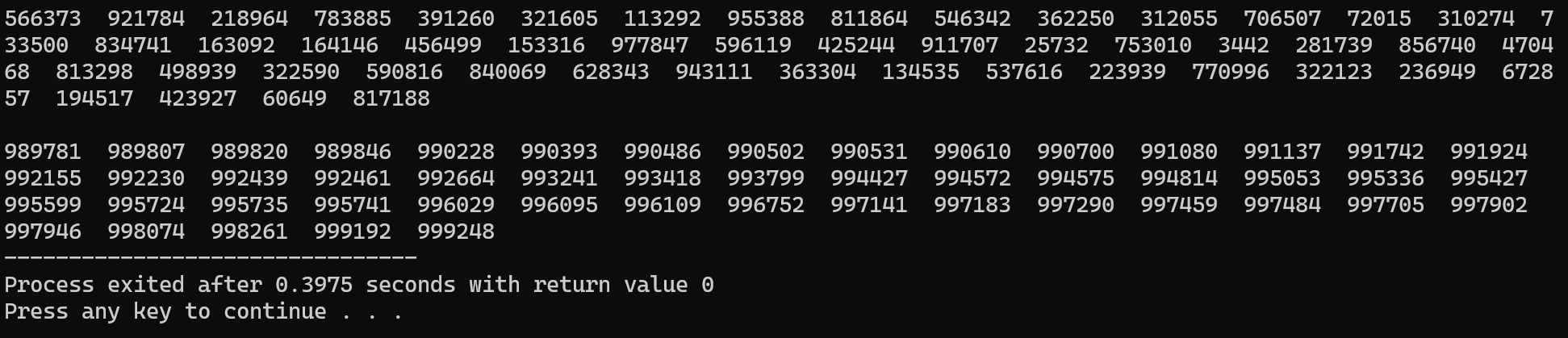
}

free(arr);

return 0;

}

OUTPUT:



OBJECTIVE: Write a program to implement bubble sort with flag.

PROGRAM:

#include<stdio.h>

#include <stdlib.h>

#define NUM\_NUMBERS 5000

void interchange(int a[],int i,int j)

{

int p=a[i];

a[i]=a[j];

a[j]=p;

}

void bubbleSort(int arr[], int n) {

int i,j,flag;

for (i = 0; i < NUM\_NUMBERS-1; i++) {

flag=0;

for (j = 0; j < NUM\_NUMBERS- i-1 ; j++) {

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

{

interchange(arr, j, j + 1);

flag=1;

}

}

if (flag == 0)

break;

}

}

int main()

{

int i;

int \*arr=(int \*)malloc(NUM\_NUMBERS \*sizeof(int));

const char \*file\_path="D:\\7th semester\\random\_numbers.txt";

FILE \*file=fopen(file\_path,"r");

for(i=0;i<NUM\_NUMBERS;i++)

{

if(fscanf(file,"%d",&arr[i])!=1)

{

printf("Error\n");

fclose(file);

}

}

for(i=NUM\_NUMBERS-50;i<NUM\_NUMBERS;i++)

{

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

}

printf("\n\n");

bubbleSort(arr,NUM\_NUMBERS);

for(i=NUM\_NUMBERS-50;i<NUM\_NUMBERS;i++)

{

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

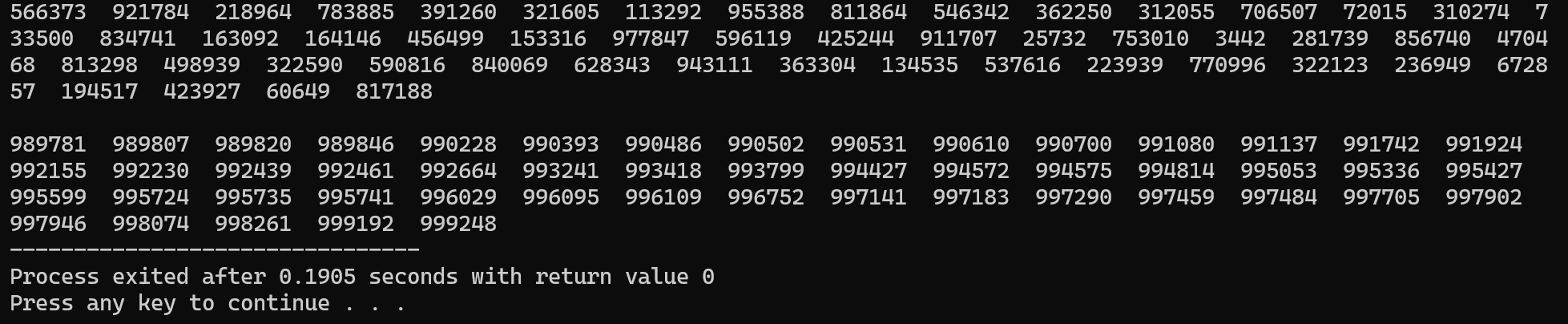
}

free(arr);

return 0;

}

OUTPUT:



OBJECTIVE:Write a program to implement binary search algorithm.

PROGRAM:

#include <stdio.h>

int binarySearch(int arr[], int size, int target) {

int low = 0;

int high = size - 1;

while (low <= high) {

int mid = (low + high) / 2;

if (arr[mid] == target) {

return mid;

} else if (arr[mid] < target) {

low = mid + 1;

} else {

high = mid - 1;

}

}

return -1;

}

int main() {

int size, target;

printf("Enter the number of elements in the array: ");

scanf("%d", &size);

int arr[size];

printf("Enter %d elements (sorted in ascending order):\n", size);

for (int i = 0; i < size; i++) {

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

}

printf("Enter the element to search: ");

scanf("%d", &target);

int result = binarySearch(arr, size, target);

if (result != -1) {

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

} else {

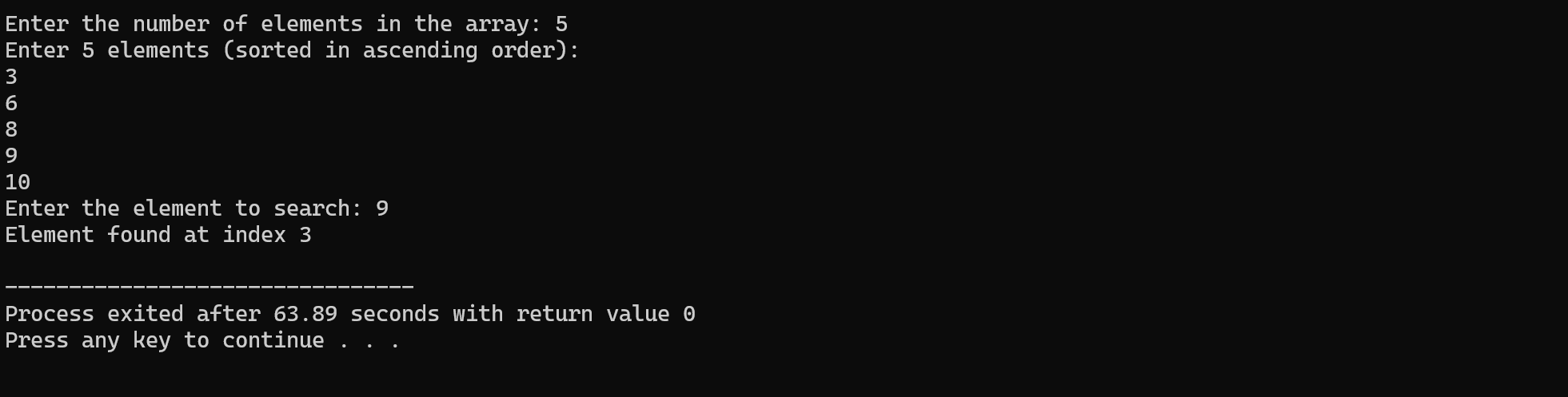
printf("Element not found in the array\n");

}

return 0;

}

OUTPUT:



OBJECTIVE:Write a program to implement merge sort algorithm.

PROGRAM:

#include<stdio.h>

#include <stdlib.h>

#define NUM\_NUMBERS 100000

void merge(int a[],int low,int mid,int high)

{

int \*b=(int \*)malloc(NUM\_NUMBERS \*sizeof(int));

if(b == NULL)

printf("Memory not allocated");

int k=low,i=low,j=mid+1;

while(i <= mid && j <= high)

{

if(a[i] <= a[j])

{

b[k++]=a[i++];

}

else

{

b[k++]=a[j++];

}

}

while(i <= mid)

b[k++]=a[i++];

while(j <= high)

b[k++]=a[j++];

for(i = low;i <= high; i++)

a[i]=b[i];

free(b);

}

void mergesort(int A[],int low,int high)

{

if(low < high)

{

int mid=(low + high) / 2;

mergesort(A,low,mid);

mergesort(A,mid + 1,high);

merge(A,low,mid,high);

}

}

int main()

{

int i;

int \*arr=(int \*)malloc(NUM\_NUMBERS \*sizeof(int));

const char \*file\_path="D:\\7th semester\\rand\_nm.txt";

FILE \*file=fopen(file\_path,"r");

for(i=0;i<NUM\_NUMBERS;i++)

{

if(fscanf(file,"%d",&arr[i])!=1)

{

printf("Error\n");

fclose(file);

}

}

for(i=NUM\_NUMBERS-50;i<NUM\_NUMBERS;i++)

{

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

}

printf("\n\n");

mergesort(arr,0,NUM\_NUMBERS-1);

for(i=NUM\_NUMBERS-50;i<NUM\_NUMBERS;i++)

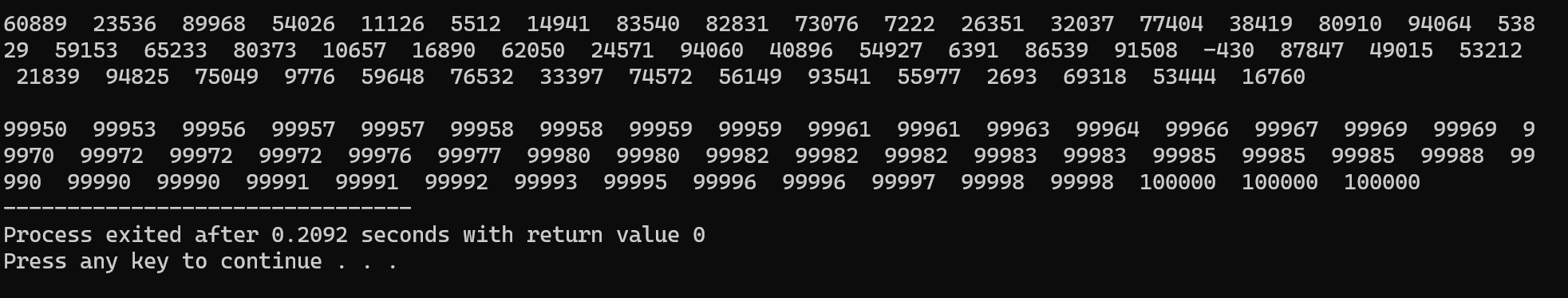
{

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

}

return 0;

}



OBJECTIVE: Write a program to implement Quick sort algorithm.

PROGRAM:

#include<stdio.h>

#include <stdlib.h>

#define NUM\_NUMBERS 10000000

void interchange(int a[],int i,int j)

{

int p=a[i];

a[i]=a[j];

a[j]=p;

}

int partition(int a[],int low,int high)

{

int pivot=a[low],i=low,j=high;

do

{

do

{

i++;

}while(a[i]<pivot && (i<=high));

do

{

j--;

}while(a[j]>pivot && (j>=low));

if(i<j)

{

interchange(a,i,j);

}

}while(i<j);

a[low]=a[j];

a[j]=pivot;

return j;

}

void quicksort(int a[],int low,int high)

{

int j;

if(low<high)

{

//printf("k");

j=partition(a,low,high+1);

quicksort(a,low,j-1);

//printf("''''''''''''''''''");

quicksort(a,j+1,high);

//printf("y");

}

}

int main()

{

int i;

int \*arr=(int \*)malloc(NUM\_NUMBERS \*sizeof(int));

const char \*file\_path="D:\\7th semester\\rand\_nm.txt";

FILE \*file=fopen(file\_path,"r");

for(i=0;i<NUM\_NUMBERS;i++)

{

if(fscanf(file,"%d",&arr[i])!=1)

{

printf("Error\n");

fclose(file);

}

}

for(i=NUM\_NUMBERS-50;i<NUM\_NUMBERS;i++)

{

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

}

printf("\n\n");

quicksort(arr,0,NUM\_NUMBERS-1);

for(i=NUM\_NUMBERS-50;i<NUM\_NUMBERS;i++)

{

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

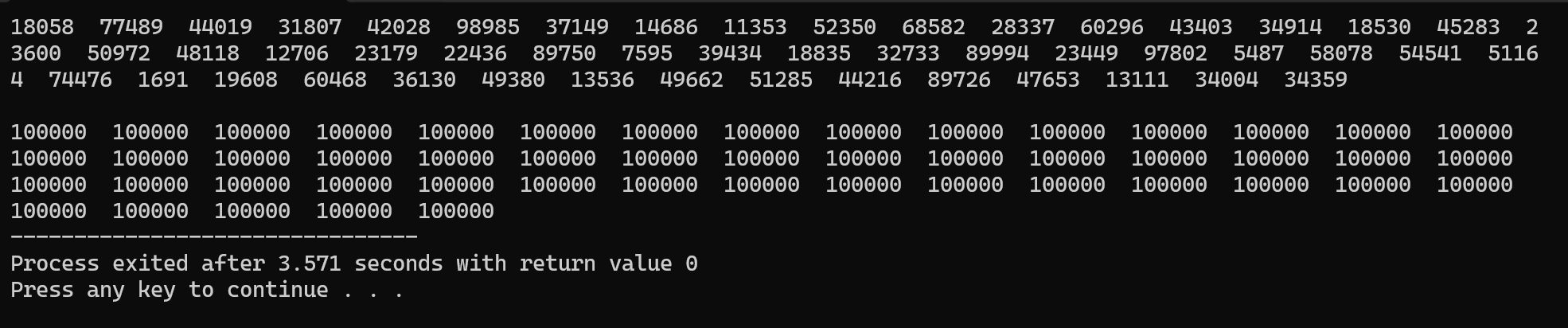
}

free(arr);

return 0;

}

OUTPUT:



OBJECTIVE: Write a program to implement non deterministic knapsack algorithm.

PROGRAM:

#include<stdio.h>

#include<stdlib.h>

int main()

{

int n,M,k,i;

int profit=0,weight=0;

printf("Enter the number of elements \n");

scanf("%d",&n);

int w[n],p[n],x[n];

printf("Enter the weights of elements\n");

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

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

printf("Enter the profits of elements\n");

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

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

printf("Enter capacity of knapsack\n");

scanf("%d",&M);

printf("Enter minimum profit value\n");

scanf("%d",&k);

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

x[i]=rand()%2;

printf("Solution vector is \n");

printf("[");

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

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

printf("]\n");

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

{

profit+=x[i]\*p[i];

weight+=x[i]\*w[i];

}

if(weight>M|| profit<k)

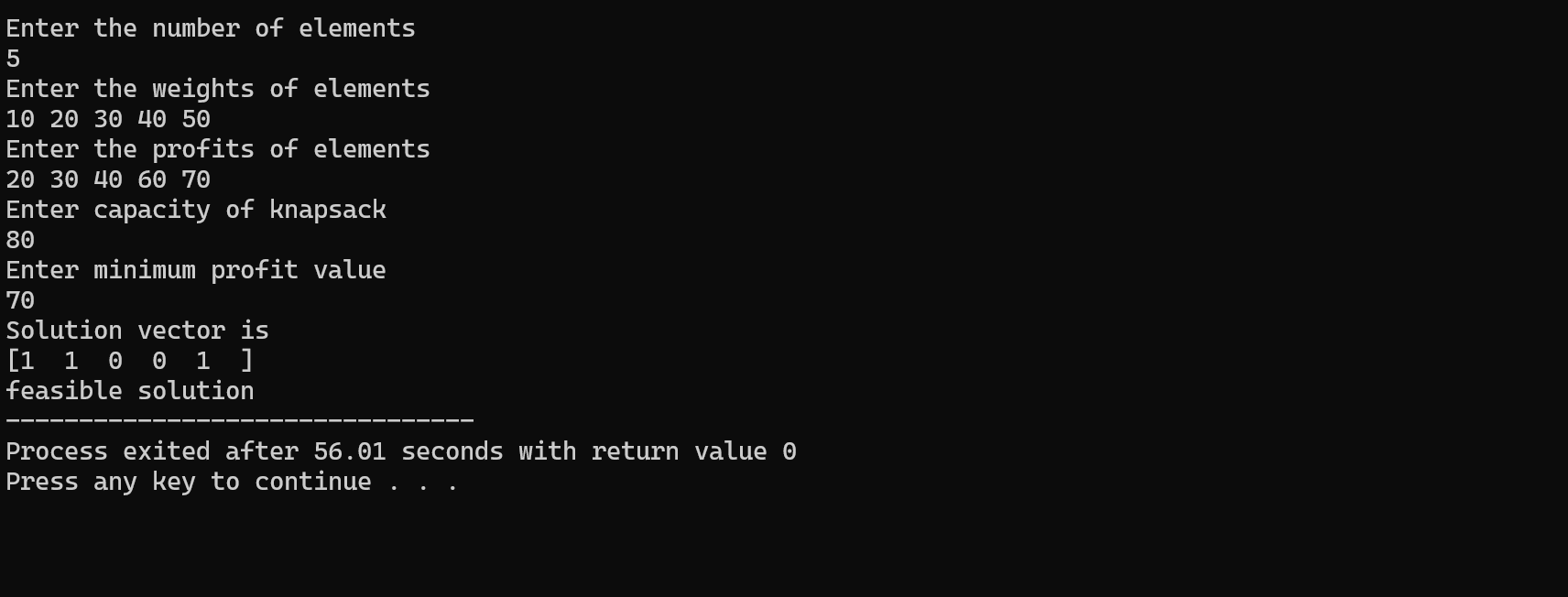
printf("Not feasible solution");

else

printf("feasible solution");

return 0;

}



OBJECTIVE: Write a program to implement non deterministic linear search algorithm.

PROGRAM:

#include<stdio.h>

#include <stdlib.h>

int ndlinear(int arr[],int size,int x)

{

int res=-1;

int j=rand( ) %size;

printf("Random generated index is %d\n",j);

if(arr[j]==x)

return j+1;

return res;

}

int main()

{

int i,n,x,result;

printf("Enter array size : ");

scanf("%d",&n);

int a[n];

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

{

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

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

}

printf("Enter element you want to search in given array \n");

scanf("%d",&x);

result=ndlinear(a,n,x);

if(result!=-1)

printf("Element %d found at %d position \n",x,result);

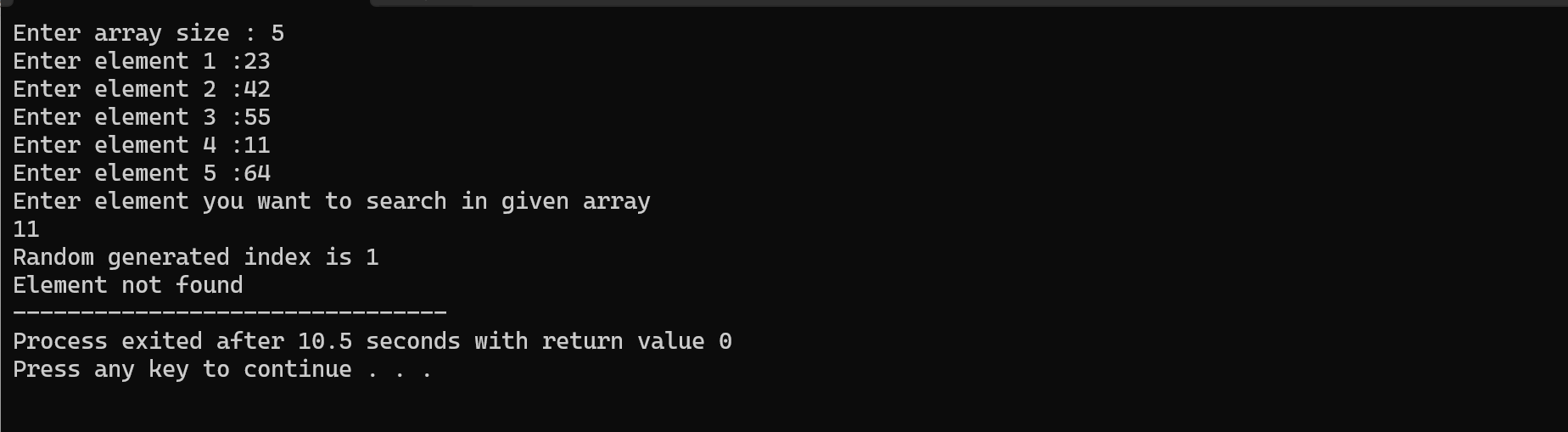
else

printf("Element not found");

return 0;

}

OUTPUT:



OBJECTIVE:WAP Primality checking

PROGRAM:

#include<stdio.h>

#include<stdlib.h>

int ptesting(int n,int k)

{

int i=1,remainder,r;

L:

r=(rand()%(n-2))+2;

remainder=n%r;

if(remainder==0)

goto out;

else

i++;

if(i<=k)

goto L;

printf("No is Prime ");

return 0;

out:

printf("No is Composite");

return 0;

}

int main()

{

int N,k;

printf("Enter a Number\n");

scanf("%d",&N);

printf("Enter minimum elements you want to check\n");

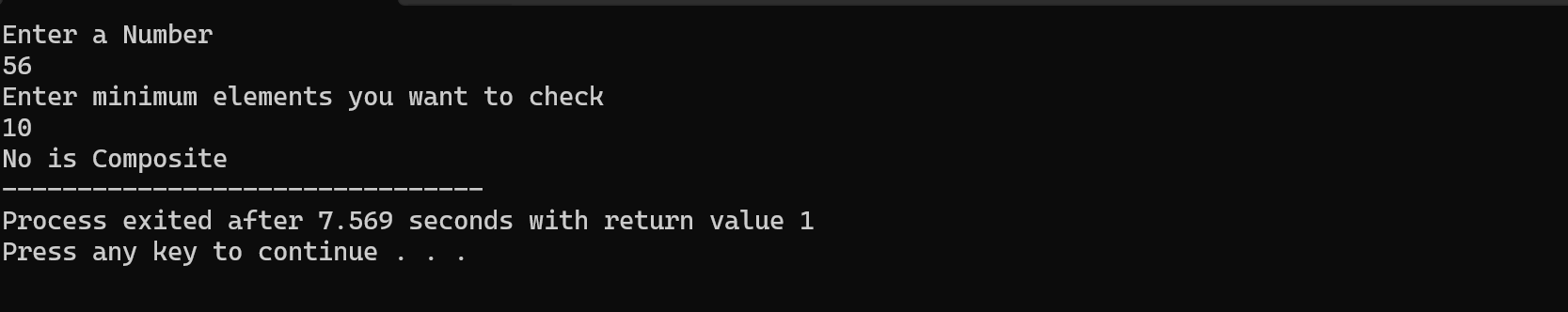
scanf("%d",&k);

ptesting(N,k);

return 1;

}

OUTPUT:



OBJECTIVE:WAP sumofsubset

PROGRAM:

#include<stdio.h>

#include<stdlib.h>

int w[6],x[6]={0},m,r=0,n=6;

void print(int x[],int n)

{

int i;

printf("[");

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

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

printf("]\n");

printf("Subsets are [");

for(int i=0;i<6;i++)

{

if(x[i]==1)

{

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

}

}

printf("]\n");

}

void sumofsubset(int s,int k,int r)

{

int i;

if(k>=n)

return ;

x[k]=1;

if(s+w[k]==m)

{

print(x,n);

}

else if( k+1<n && s+w[k]+w[k+1]<=m)

sumofsubset(s+w[k],k+1,r-w[k]);

x[k]=0;

if(s+r-w[k]>=m && k+1 < n && s+w[k+1]<=m)

{

sumofsubset(s,k+1,r-w[k]);

}

}

int main()

{

printf("Enter weights of elements: ");

for(int i=0;i<6;i++)

{

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

r+=w[i];

}

printf("Enter the sum ");

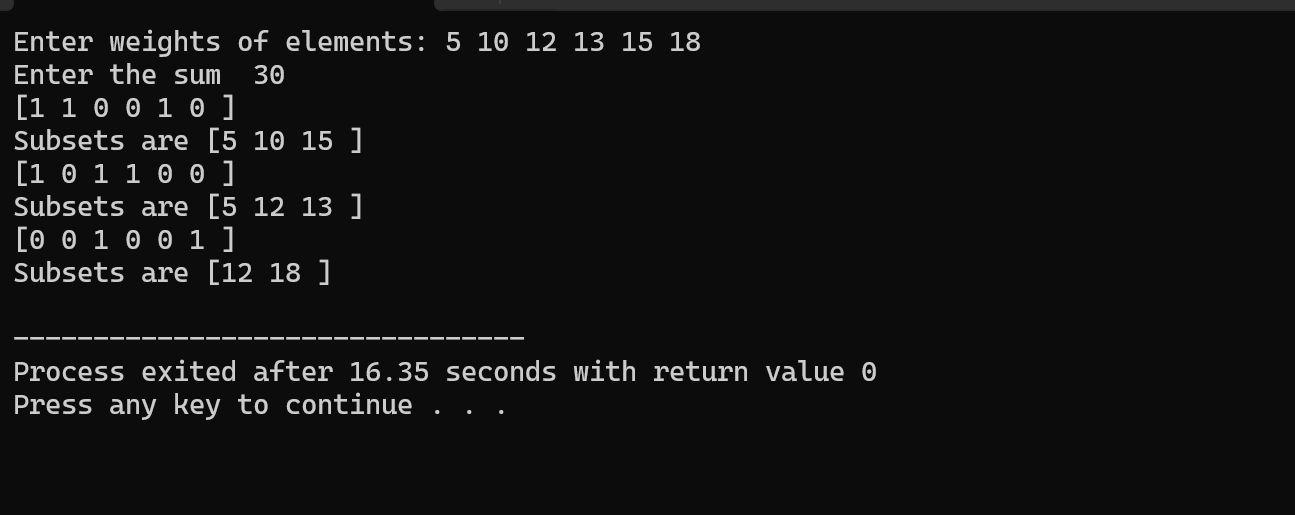
scanf("%d",&m);

//print(w,n);

sumofsubset(0,0,r);

}

OUTPUT:



OBJECTIVE:counting sort

PROGRAM:

#include <stdio.h>

int findMax(int A[], int n) {

int max = A[0];

for (int i = 1; i < n; i++) {

if (A[i] > max) {

max = A[i];

}

}

return max;

}

void countingSort(int A[], int n, int k) {

int B[n];

int C[k + 1];

for (int i = 0; i <= k; i++) {

C[i] = 0;

}

for (int j = 0; j < n; j++) {

C[A[j]]++;

}

for (int i = 1; i <= k; i++) {

C[i] += C[i - 1];

}

for (int j = n - 1; j >= 0; j--) {

B[C[A[j]] - 1] = A[j];

C[A[j]]--;

}

for (int i = 0; i < n; i++) {

A[i] = B[i];

}

}

int main() {

int n;

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

scanf("%d", &n);

int A[n];

printf("Enter the elements of the array: ");

for (int i = 0; i < n; i++) {

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

}

int k = findMax(A, n);

countingSort(A, n, k);

printf("Sorted array: ");

for (int i = 0; i < n; i++) {

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

}

return 0;

}

OUTPUT:

