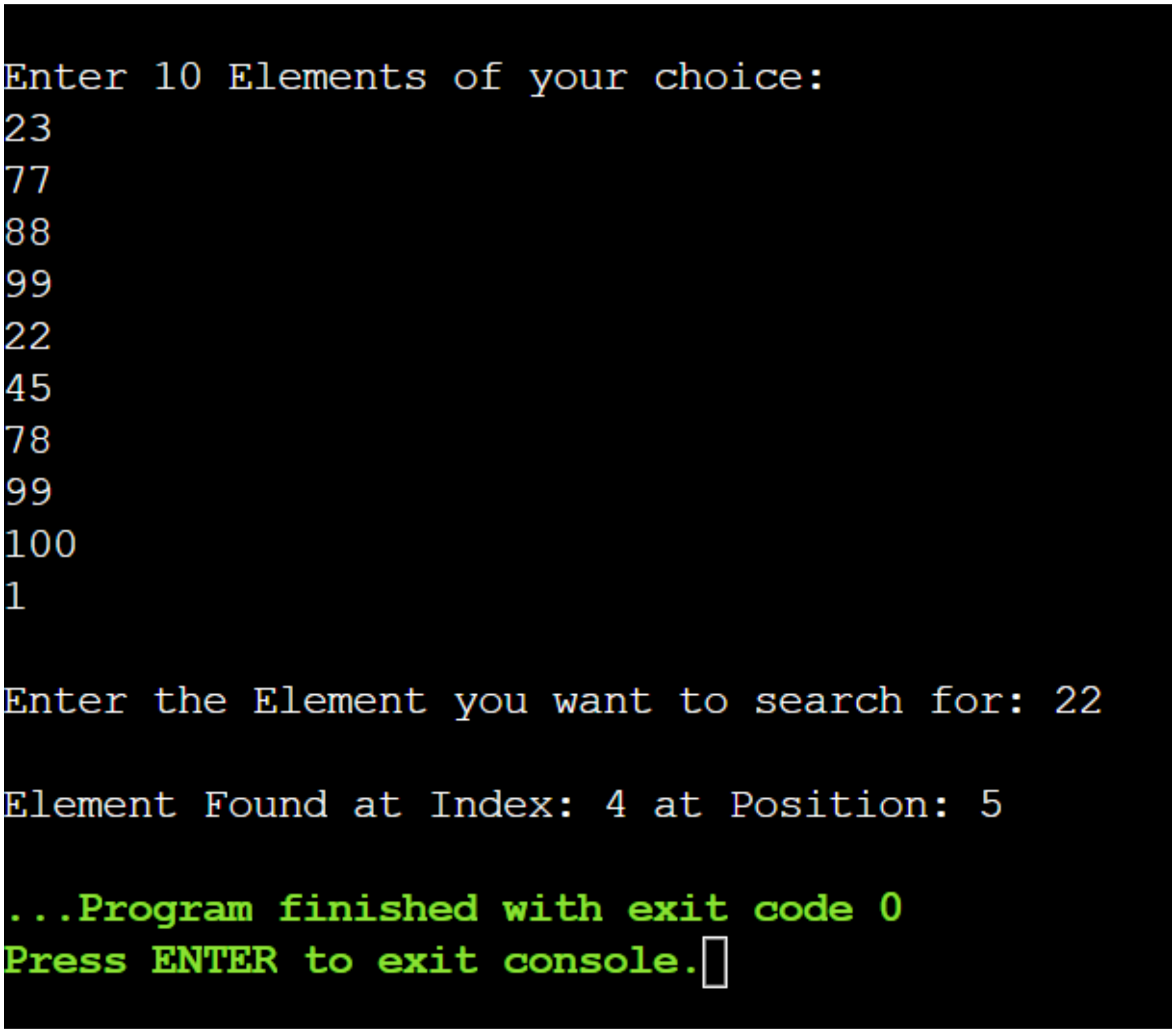
10. Write a program to implement linear search.

Code :

```
#include <iostream>
using namespace std;
//Linear Search

void linearSearch(int a[], int x)
{
int temp = -1;
for(int i=0; i<10; i++)
{
if(a[i]==x)
{
cout<<"\nElement Found at Index: "<<i<<" at Position: "<<i+1;
temp = 0;
}
}
if(temp==-1)
{
cout<<"\nElement Not Found...";
}
}

int main(){

int arr[10],n,a;
cout<<"\nEnter 10 Elements of your choice: "<<endl; for(int i=0; i<10; i++)
{
cin>>arr[i];
}
cout<<"\nEnter the Element you want to search for: "; cin>>a;

linearSearch(arr,a);

return 0;
}
```

EN20CS301486 Vaishnavi Kag

Output :



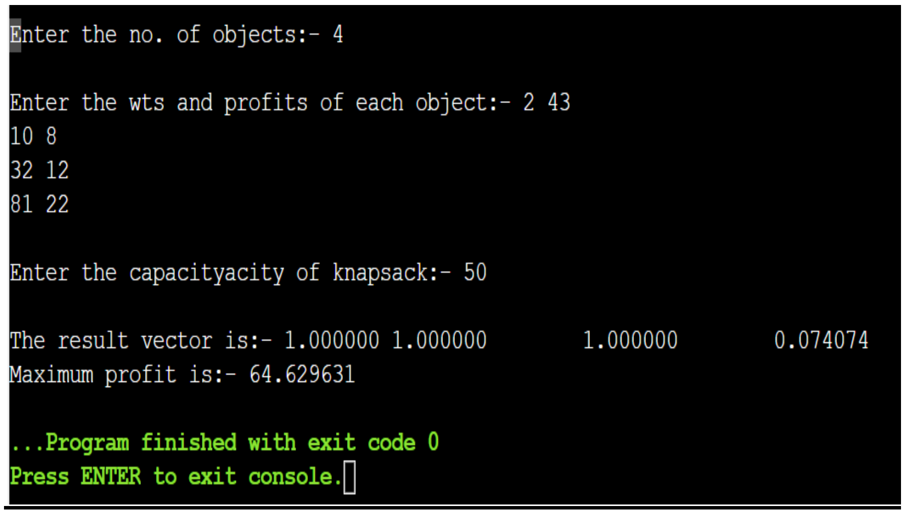
```
Enter 10 Elements of your choice:
23
77
88
99
22
45
78
99
100
1

Enter the Element you want to search for: 22

Element Found at Index: 4 at Position: 5

...Program finished with exit code 0
Press ENTER to exit console.
```

9.Write a program to implement Knapsack problem using Greedy approach.

Code:

```
# include<stdio.h>

void knapsack(int n, float weight[], float profit[], float capacity) {
  float x[20], tp = 0;
  int i, j, u;
  u = capacity;

  for (i = 0; i < n; i++)
    x[i] = 0.0;

  for (i = 0; i < n; i++) {
    if (weight[i] > u)
      break;
    else {
      x[i] = 1.0;
      tp = tp + profit[i];
      u = u - weight[i];
    }
  }

  if (i < n)
    x[i] = u / weight[i];

  tp = tp + (x[i] * profit[i]);

  printf("\nThe result vector is:- ");
  for (i = 0; i < n; i++)
    printf("%f\t", x[i]);

  printf("\nMaximum profit is:- %f", tp);

}

int main() {
  float weight[20], profit[20], capacity;
  int num, i, j;
  float ratio[20], temp;

  printf("\nEnter the no. of objects:- ");
  scanf("%d", &num);

  printf("\nEnter the wts and profits of each object:- ");
  for (i = 0; i < num; i++) {
    scanf("%f %f", &weight[i], &profit[i]);
  }
  printf("\nEnter the capacityacity of knapsack:- ");
  scanf("%f", &capacity);

  for (i = 0; i < num; i++) {
    ratio[i] = profit[i] / weight[i];
  }

  for (i = 0; i < num; i++) {
    for (j = i + 1; j < num; j++) {
      if (ratio[i] < ratio[j]) {
        temp = ratio[j];
        ratio[j] = ratio[i];
        ratio[i] = temp;

        temp = weight[j];
        weight[j] = weight[i];
        weight[i] = temp;

        temp = profit[j];
        profit[j] = profit[i];
        profit[i] = temp;
      }
    }
  }

  knapsack(num, weight, profit, capacity);
  return(0);
}
```

Output



```
Enter the no. of objects:- 4

Enter the wts and profits of each object:- 2 43
10 8
32 12
81 22

Enter the capacityacity of knapsack:- 50

The result vector is:- 1.000000 1.000000        1.000000        0.074074
Maximum profit is:- 64.629631

...Program finished with exit code 0
Press ENTER to exit console.
```