Assignment 11

class - SE IV Roll NO - 21430
Batch - F4
D.O.S - 26/11/2020

Problem Statement:
Write C++ program using STL for sorting and searching user defined records such as personal records (Name, DOB, Telephone NO etc) using vector container. OR.
Write C++ program using STL for sorting and searching user defined records such as item records(item, code, name, cost, quantity etc) using vector container.

Objectives:
To learn the concept of STL, searching, sorting and vector container.

Theory:
The standard Template Library (STL) is a set of C++ template classes to provide common programming data structures and functions such as lists, stack, arrays etc.
It is library of container classes, algorithm and iterators. It is generalized library and so, its components are parameterized. A working

knowledge of template classes is a prerequisite for working with STL.

STL has four components
i) Algorithms ii) Containers
iii) functions iv) Iterators

i) Algorithms
The algorithm defines a collection of function especially designed to be used to on ranges of elements. They act an containes and provides means for various operations for the contents of the containers.

ii) Container -:
Containers or container classes store object and data. There are in total 7 standard containers classes and 3 container adapter classes and only 7 header files that provide class access to these container or container adapters.

1. Sequence Container -
Implement data structure which can be accessed in sequential manner.
- vector - deque
- list - arrays.
- forward list (Introduced in C++)

2. Container Adaptor -
Provide different interface for sequential containers.
- queque - stack
- priority-queue

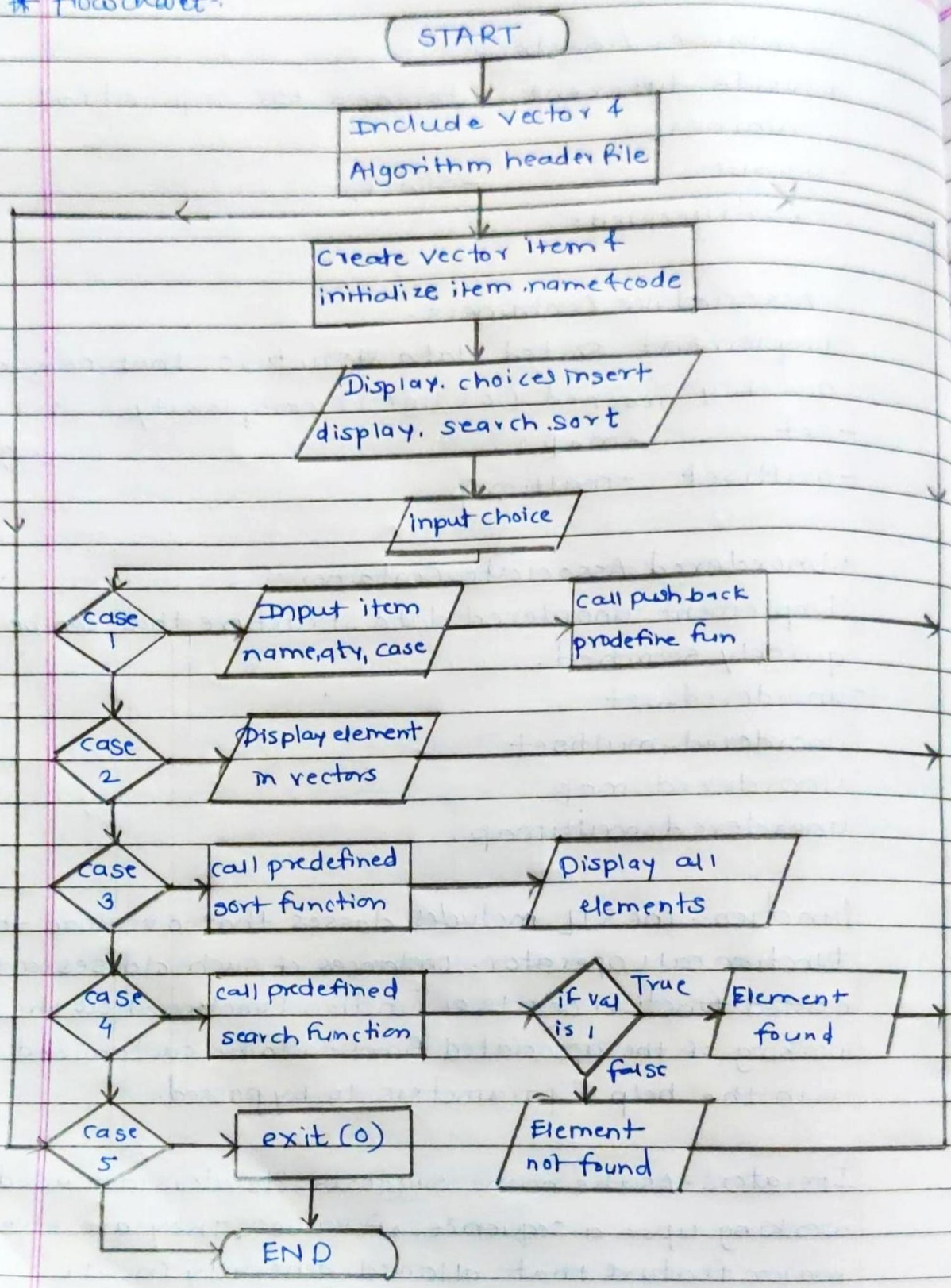
3. Associative containers.
Implement sorted data structure that can be quickly reached (o(lgn)) complexity.
- set - map
- multiset - multimap.

4. Unordered Associate Container :
Implement unordered data structure that can be quickly searched.
- unordered_set
unordered-multiset
unordered_map
unordered_multimap.

function : The STL includes classes that overload the function call operator. Instances of such classes are called function objects of function Functors allow the working of the associated function to be customized with the help of parameters to by passed.

Iterators - As the name suggests, iterators are used for working upon a sequence of values. They are the major feature that allowed generally in STL

* Flowchart:-


START
Include vector & Algorithm header file
Create vector item & initialize item.name&code
Display. choices insert display. search. sort
input choice
case 1
Input item name,qty, case
call push back predefine fun
case 2
Display element in vectors
case 3
call predefined sort function
Display all elements
case 4
call predefined search function
if val is 1
True
Element found
false
Element not found
case 5
exit (0)
END

Test Cases.

NO.	Description	Input	Expected O/P.	Actual O/P	Result
1.	Menu -: 1. Insert 2. Display 3. sort 4. search 5. Exit	choice: 1 Name: Abc qty - 5 price - 45 code - 7	Name. Abc qty - 5 price. 45 code - 7	Name. Abc qty - 5 price. 45 code - 7	Pass.
2.	Menu 1. Insert 2. Display 3. sort 4. Search 5. Exit	Choice: 4 Enter code: 4	Not found	Not found	Pass

Conclusion -:

We learnt to implement concept of STL. Searching sorting and vector containers.

```cpp
#include <iostream>
#include <algorithm>
#include <vector>

using namespace std;
class Item
{
    public:
        char name[10];
        int quantity;
        int cost;
        int code;

        bool operator==(const Item& i1)
    {
        if(code==i1.code)

        return 1;

        return 0;
    }

    bool operator<(const Item& i1)
    {
        if(code<i1.code)

        return 1;

        return 0;
    }

};
```

```
34  vector<Item> o1;
35  void print(Item &i1);
36  void display();
37  void insert();
38  void search();
39  void dlt();
40
41  bool compare(const Item &i1, const Item &i2)
42  {
43
44     return i1.cost < i2.cost;
45  }
46
47  int main()
48  {
49      int ch;
50        do
51      {
52          cout<<"\n Menu ";
53          cout<<"\n1.Insert";
54          cout<<"\n2.Display";
55          cout<<"\n3.Search";
56          cout<<"\n4.Sort";
57          cout<<"\n5.Delete";
58          cout<<"\n6.Exit";
59          cout<<"\nchoice:";
60          cin>>ch;
61
62          switch(ch)
63          {
64                  case 1:
65                                  insert();
```

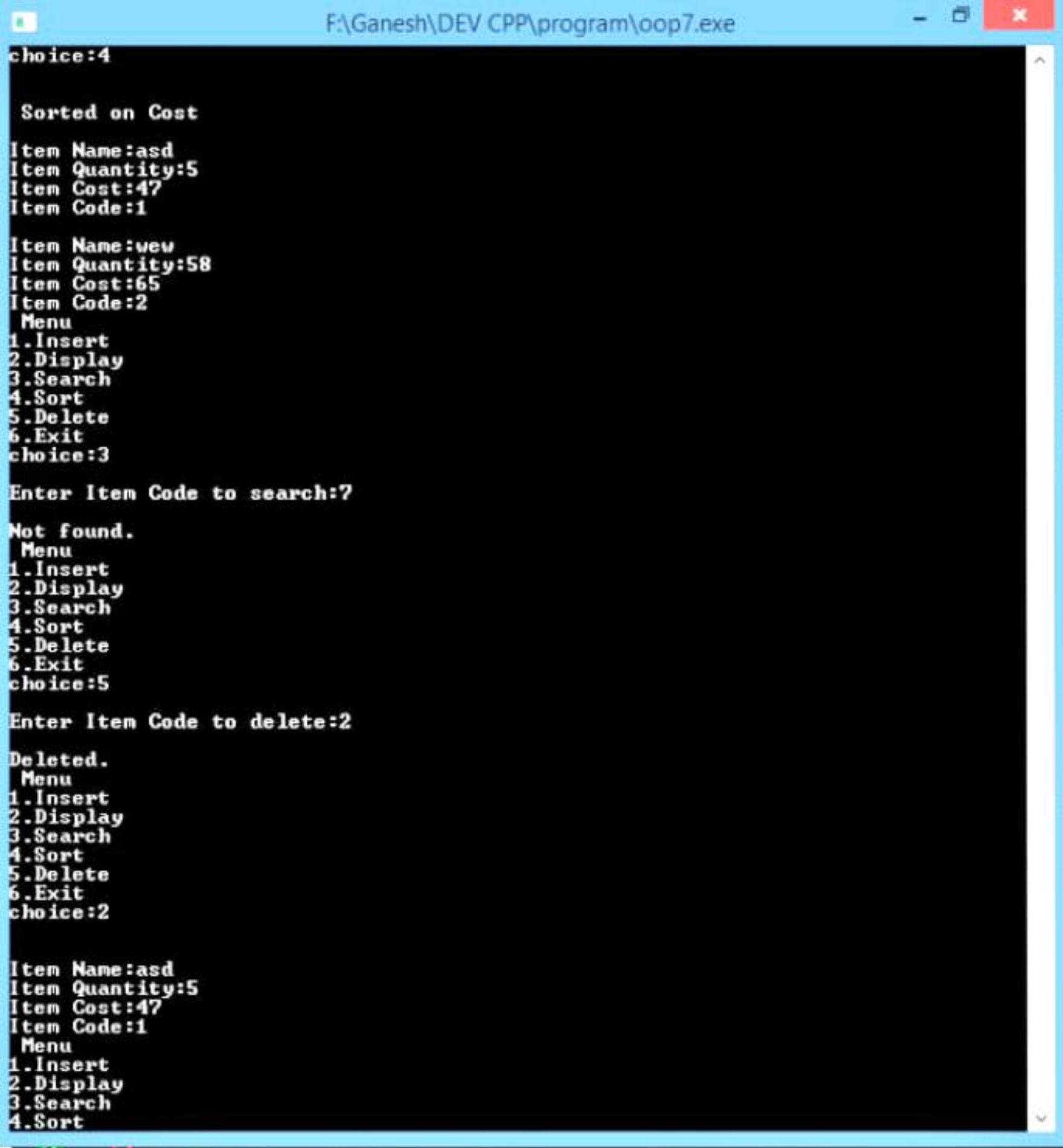
```
67
68                  case 2:
69                      display();
70                      break;
71
72                  case 3:
73                      search();
74                      break;
75
76                  case 4:
77                      sort(o1.begin(),o1.end(),compare);
78                      cout<<"\n\n Sorted on Cost";
79                      display();
80                      break;
81
82                  case 5:
83                     dlt();
84                      break;
85
86                  case 6:
87                      exit(0);
88          }
89
90      }while(ch!=7);
91
92        return 0;
93  }
94
95  void insert()
96  {
97      Item i1;
98      cout<<"\nEnter Item Name:";
```

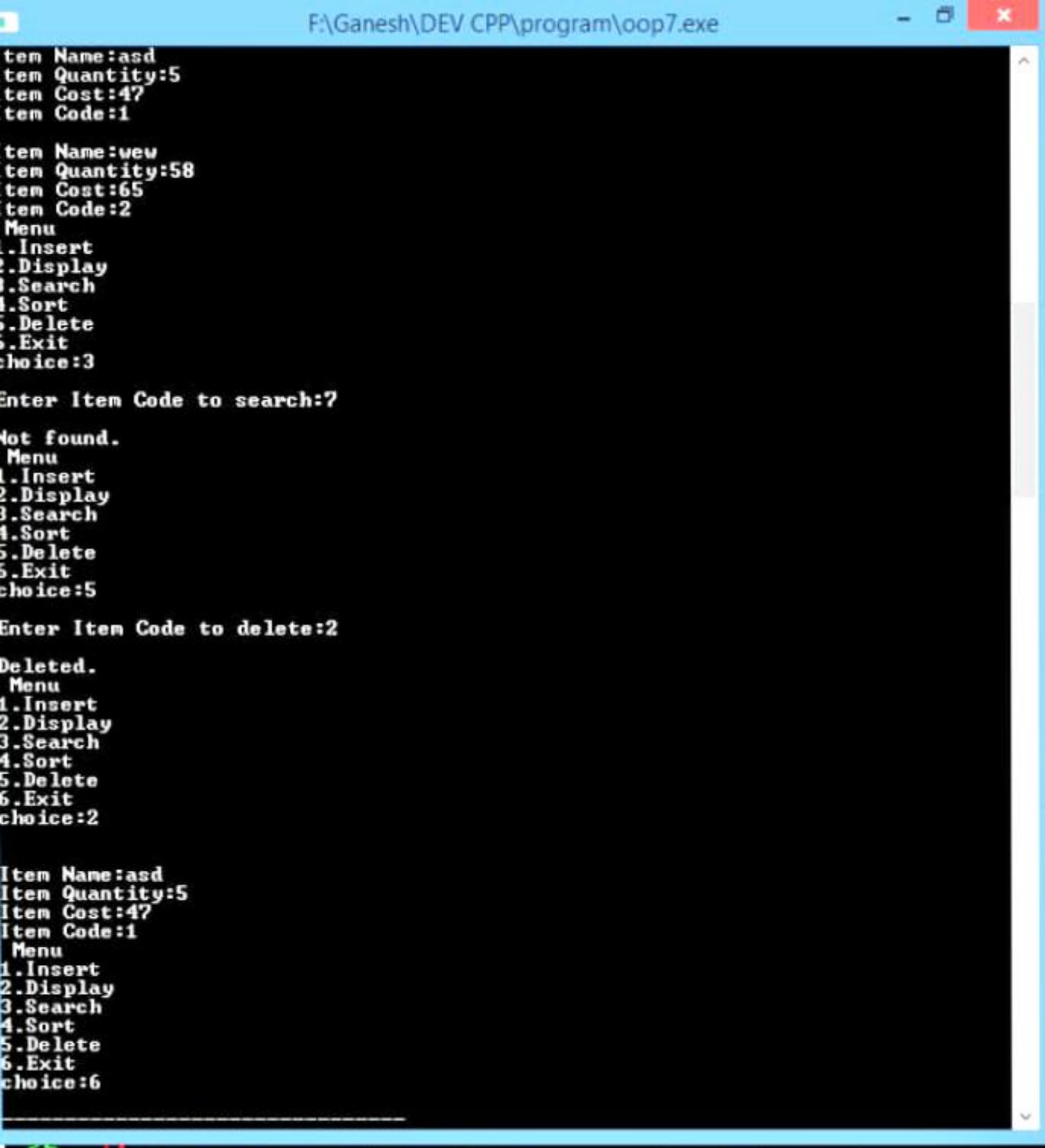
```
{
    Item i1;
    cout<<"\nEnter Item Name:";
    cin>>i1.name;
    cout<<"\nEnter Item Quantity:";
    cin>>i1.quantity;
    cout<<"\nEnter Item Cost:";
    cin>>i1.cost;
    cout<<"\nEnter Item Code:";
    cin>>i1.code;
    o1.push_back(i1);
}

void display()
{
    for_each(o1.begin(),o1.end(),print);
}

void print(Item &i1)
{
    cout<<"\n";
    cout<<"\nItem Name:"<<i1.name;
    cout<<"\nItem Quantity:"<<i1.quantity;
    cout<<"\nItem Cost:"<<i1.cost;
    cout<<"\nItem Code:"<<i1.code;
}

void search()
{
    vector<Item>::iterator p;
    Item i1;
    cout<<"\nEnter Item Code to search:";
```

```
125     vector<Item>::iterator p;
126     Item i1;
127     cout<<"\nEnter Item Code to search:";
128     cin>>i1.code;
129     p=find(o1.begin(),o1.end(),i1);
130     if(p==o1.end())
131     {
132         cout<<"\nNot found.";
133     }
134     else
135     {
136         cout<<"\nFound.";
137     }
138 }
139
140 void dlt()
141 {
142     vector<Item>::iterator p;
143     Item i1;
144     cout<<"\nEnter Item Code to delete:";
145     cin>>i1.code;
146     p=find(o1.begin(),o1.end(),i1);
147     if(p==o1.end())
148     {
149         cout<<"\nNot found.";
150     }
151     else
152     {
153         o1.erase(p);
154         cout<<"\nDeleted.";
155     }
156 }
```

F:\Ganesh\DEV CPP\program\oop7.exe

```
 Menu
1.Insert
2.Display
3.Search
4.Sort
5.Delete
6.Exit
choice:1

Enter Item Name:asd

Enter Item Quantity:5

Enter Item Cost:47

Enter Item Code:1

 Menu
1.Insert
2.Display
3.Search
4.Sort
5.Delete
6.Exit
choice:1

Enter Item Name:vew

Enter Item Quantity:58

Enter Item Cost:65

Enter Item Code:2

 Menu
1.Insert
2.Display
3.Search
4.Sort
5.Delete
6.Exit
choice:2


Item Name:asd
Item Quantity:5
Item Cost:47
Item Code:1

Item Name:vew
Item Quantity:58
Item Cost:65
Item Code:2
 Menu
1.Insert
2.Display
```


F:\Ganesh\DEV CPP\program\oop7.exe
choice:4
Sorted on Cost
Item Name:asd
Item Quantity:5
Item Cost:47
Item Code:1
Item Name:wew
Item Quantity:58
Item Cost:65
Item Code:2
Menu
1.Insert
2.Display
3.Search
4.Sort
5.Delete
6.Exit
choice:3
Enter Item Code to search:7
Not found.
Menu
1.Insert
2.Display
3.Search
4.Sort
5.Delete
6.Exit
choice:5
Enter Item Code to delete:2
Deleted.
Menu
1.Insert
2.Display
3.Search
4.Sort
5.Delete
6.Exit
choice:2
Item Name:asd
Item Quantity:5
Item Cost:47
Item Code:1
Menu
1.Insert
2.Display
3.Search
4.Sort


F:\Ganesh\DEV CPP\program\oop7.exe
tem Name:asd
tem Quantity:5
tem Cost:47
tem Code:1
tem Name:wew
tem Quantity:58
tem Cost:65
tem Code:2
Menu
.Insert
.Display
.Search
.Sort
.Delete
.Exit
choice:3
Enter Item Code to search:7
Not found.
Menu
.Insert
.Display
.Search
.Sort
.Delete
.Exit
choice:5
Enter Item Code to delete:2
Deleted.
Menu
.Insert
.Display
.Search
.Sort
.Delete
.Exit
choice:2
Item Name:asd
Item Quantity:5
Item Cost:47
Item Code:1
Menu
.Insert
.Display
.Search
.Sort
.Delete
.Exit
choice:6