Experiment No. : 7

Title: Hash Table implementation

Batch:A2 **Roll No.:16010421063** **Experiment No.: 7**

Aim: Write a menu driven program to implement a hash table.

Resources Used: C/ C++ editor and compiler.

Theory:

Hash Table

Hash table is one of the most important data structures that uses a special function known as a hash function that maps a given value with a key to access the elements faster.

A Hash table is a data structure that stores some information, and the information has basically two main components, i.e., key and value. The hash table can be implemented with the help of an associative array. The efficiency of mapping depends upon the efficiency of the hash function used for mapping.

For example, suppose the key value is John and the value is the phone number, so when we pass the key value in the hash function shown as below:

Hash(key)= index;

When we pass the key in the hash function, then it gives the index.

Hash(john) = 3;

The above example adds the john at the index 3.

Hashing

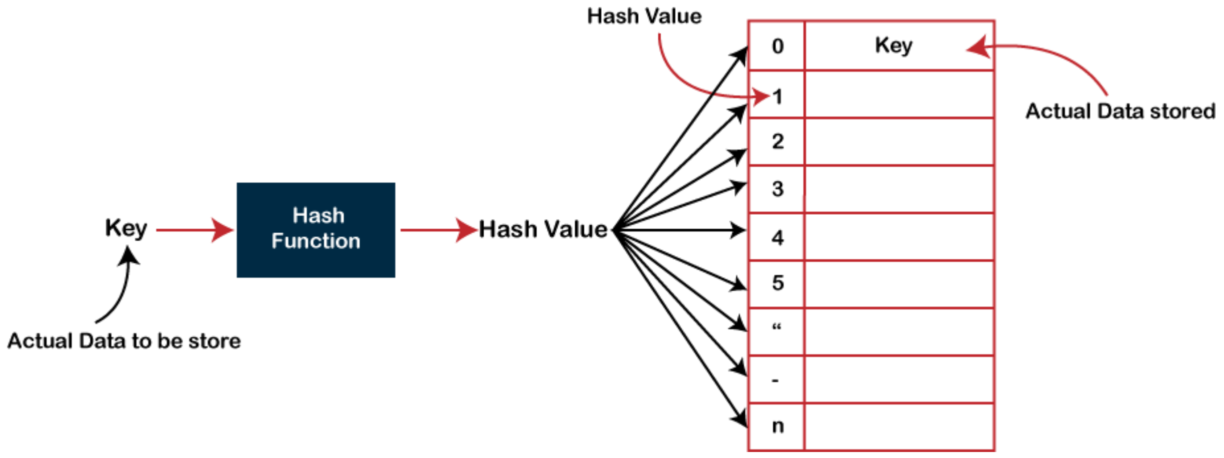
Hashing is one of the searching techniques that uses a constant time. The time complexity in hashing is O(1).

The worst time complexity in linear search is O(n), and O(logn) in binary search. In both the searching techniques, the searching depends upon the number of elements but we want the technique that takes a constant time. So, hashing technique came that provides a constant time.

In Hashing technique, the hash table and hash function are used. Using the hash function, we can calculate the address at which the value can be stored.

The main idea behind the hashing is to create the (key/value) pairs. If the key is given, then the algorithm computes the index at which the value would be stored. It can be written as:

Index = hash(key)



Drawback of Hash function

A Hash function assigns each value with a unique key. Sometimes hash table uses an imperfect hash function that causes a collision because the hash function generates the same key of two different values.

Collision

When the two different values have the same value, then the problem occurs between the two values, known as a collision. In the above example, the value is stored at index 6. If the key value is 26, then the index would be:

h(26) = 26%10 = 6

Therefore, two values are stored at the same index, i.e., 6, and this leads to the collision problem. To resolve these collisions, we have some techniques known as collision techniques.

The following are the collision techniques:

- Open Hashing: It is also known as closed addressing.
- Closed Hashing: It is also known as open addressing.

Open Hashing

- In Open Hashing, one of the methods used to resolve the collision is known as a chaining method.

Basic Operations

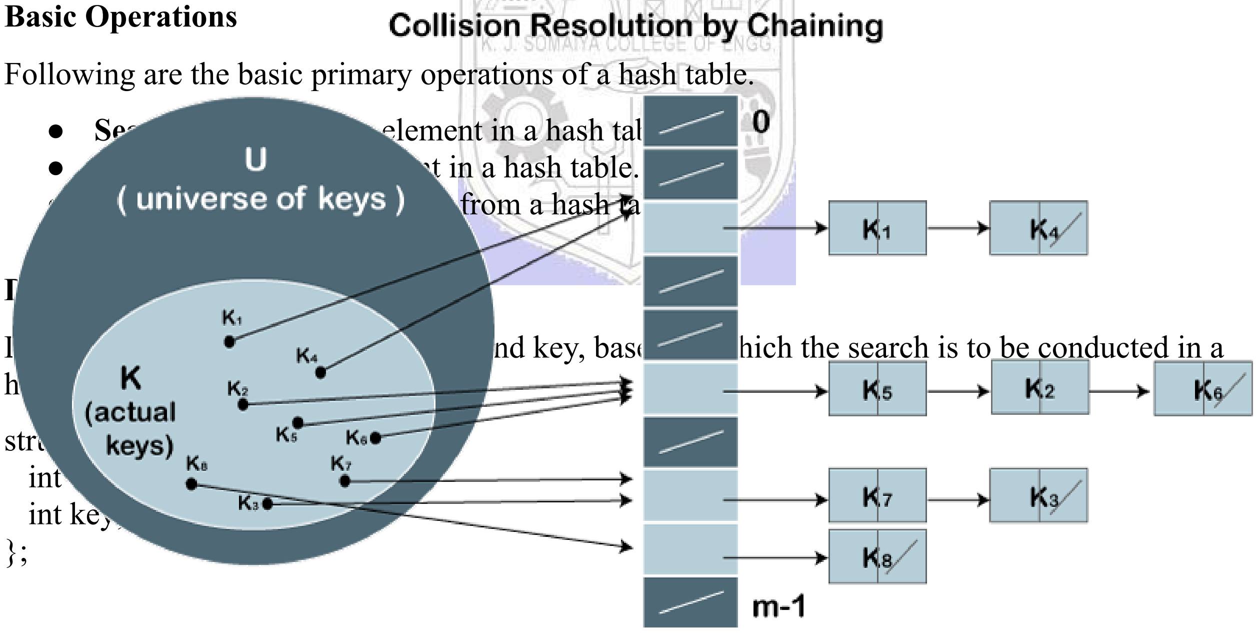
Following are the basic primary operations of a hash table.

- **Se** element in a hash tal
- t in a hash table.
- from a hash ta

D

I nd key, bas hich the search is to be conducted in a h

```
str
  int
  int key
};
```



Hash Method

Define a hashing method to compute the hash code of the key of the data item.

```
int hashCode(int key){
   return key % SIZE;
}
```

Search Operation

Whenever an element is to be searched, compute the hash code of the key passed and locate the element using that hash code as index in the array. Use linear probing to get the element ahead if the element is not found at the computed hash code.

Insert Operation

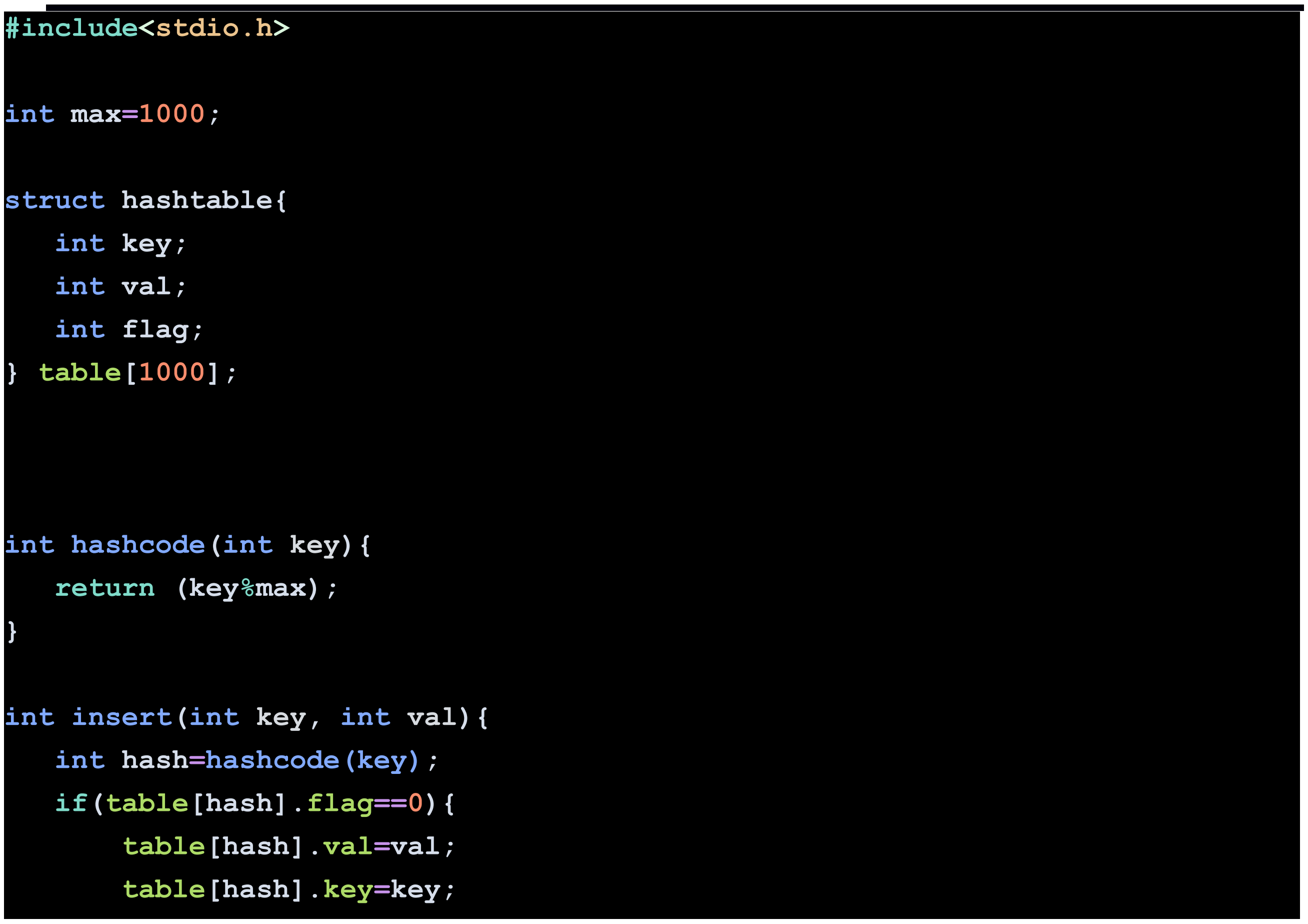
Whenever an element is to be inserted, compute the hash code of the key passed and locate the index using that hash code as an index in the array. Use linear probing for empty location, if an element is found at the computed hash code.

Delete Operation

Whenever an element is to be deleted, compute the hash code of the key passed and locate the index using that hash code as an index in the array. Use linear probing to get the element ahead if an element is not found at the computed hash code. When found, store a dummy item there to keep the performance of the hash table intact.

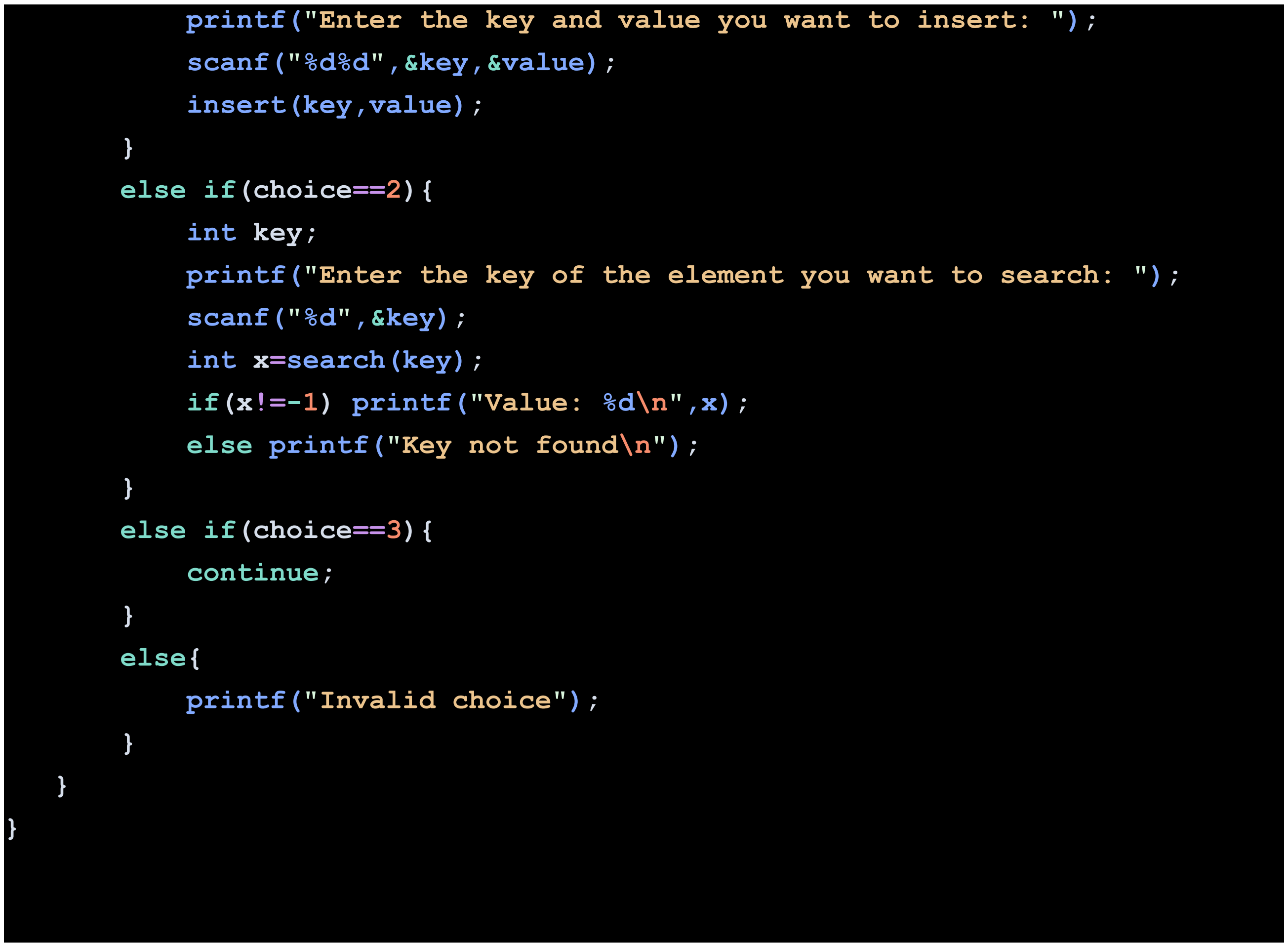
Activity: A C program depicting the searching and insertion operation in a hash table.

Results:

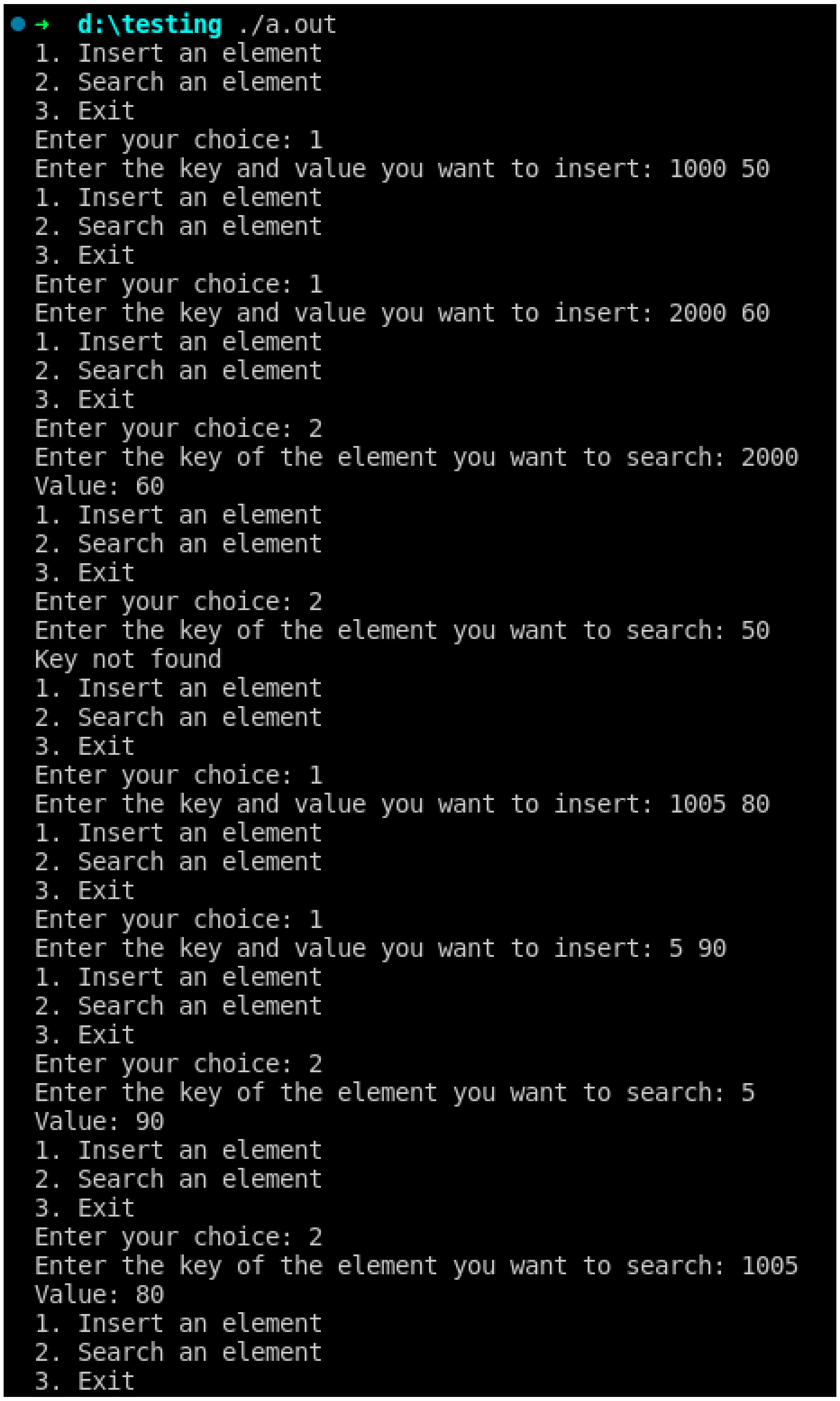


```
#include<stdio.h>

int max=1000;

struct hashtable{
   int key;
   int val;
   int flag;
} table[1000];



int hashcode(int key){
   return (key%max);
}

int insert(int key, int val){
   int hash=hashcode(key);
   if(table[hash].flag==0){
       table[hash].val=val;
       table[hash].key=key;
```

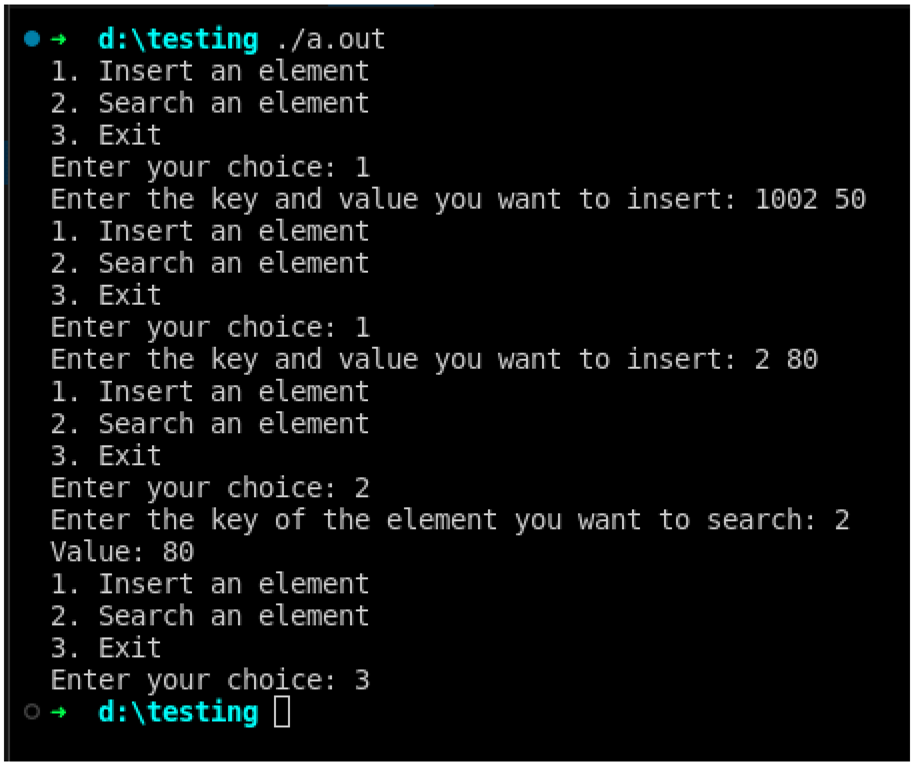
```
        table[hash].flag=1;
        return 1;
    }
    else{
        int i=1;
        while(table[(hash+1)%max].flag!=1){
            hash=(hash+1)%max;
        }
        table[(hash+i+1)%max].val=val;
        table[(hash+i+1)%max].key=key;
        table[(hash+i+1)%max].flag=1;
        return 1;
    }
}

int search(int key){
    int hash=hashcode(key);
    if(table[hash].flag==0){
        return -1;
    }
    if(table[hash].key==key){
        return table[hash].val;
    }
    else{
        int i=1;
        while(table[(hash+1)%max].key!=key){
            hash=(hash+1)%max;
        }
        return table[(hash+1)%max].val;
    }

}

int main(){

    int choice=0;
    while(choice!=3){
        printf("1. Insert an element\n2. Search an element\n3. Exit\nEnter
your choice: ");
        scanf("%d",&choice);
        if(choice==1){
            int key,value;
```



```
            printf("Enter the key and value you want to insert: ");
            scanf("%d%d",&key,&value);
            insert(key,value);
        }
        else if(choice==2){
            int key;
            printf("Enter the key of the element you want to search: ");
            scanf("%d",&key);
            int x=search(key);
            if(x!=-1) printf("Value: %d\n",x);
            else printf("Key not found\n");
        }
        else if(choice==3){
            continue;
        }
        else{
            printf("Invalid choice");
        }
    }
}
```



```
● ➜  d:\testing ./a.out
 1. Insert an element
 2. Search an element
 3. Exit
 Enter your choice: 1
 Enter the key and value you want to insert: 1000 50
 1. Insert an element
 2. Search an element
 3. Exit
 Enter your choice: 1
 Enter the key and value you want to insert: 2000 60
 1. Insert an element
 2. Search an element
 3. Exit
 Enter your choice: 2
 Enter the key of the element you want to search: 2000
 Value: 60
 1. Insert an element
 2. Search an element
 3. Exit
 Enter your choice: 2
 Enter the key of the element you want to search: 50
 Key not found
 1. Insert an element
 2. Search an element
 3. Exit
 Enter your choice: 1
 Enter the key and value you want to insert: 1005 80
 1. Insert an element
 2. Search an element
 3. Exit
 Enter your choice: 1
 Enter the key and value you want to insert: 5 90
 1. Insert an element
 2. Search an element
 3. Exit
 Enter your choice: 2
 Enter the key of the element you want to search: 5
 Value: 90
 1. Insert an element
 2. Search an element
 3. Exit
 Enter your choice: 2
 Enter the key of the element you want to search: 1005
 Value: 80
 1. Insert an element
 2. Search an element
 3. Exit
```



```
● → d:\testing ./a.out
  1. Insert an element
  2. Search an element
  3. Exit
  Enter your choice: 1
  Enter the key and value you want to insert: 1002 50
  1. Insert an element
  2. Search an element
  3. Exit
  Enter your choice: 1
  Enter the key and value you want to insert: 2 80
  1. Insert an element
  2. Search an element
  3. Exit
  Enter your choice: 2
  Enter the key of the element you want to search: 2
  Value: 80
  1. Insert an element
  2. Search an element
  3. Exit
  Enter your choice: 3
○ → d:\testing
```

Outcomes:

CO3:Describe concepts of advanced data structures like set map dictionary

---

Conclusion: We implemented the hash table with the use of linear probing to solve the problem of collision. We tested for edge cases such as the same hash, hash index at the end of the array to prevent overflow of the array.

Grade: AA / AB / BB / BC / CC / CD /DD

Signature of faculty in-charge with date

References:

Books/ Journals/ Websites:

- Michael T. Goodrich, Roberto Tamassia, and David M. Mount. 2009. Data Structures and Algorithms in C++ (2nd. ed.). Wiley Publishing.