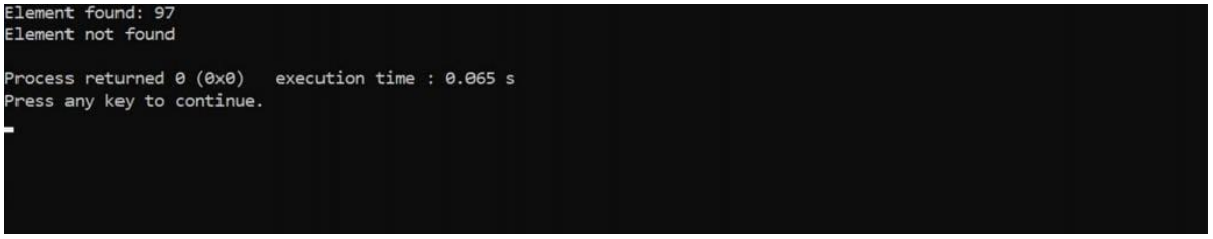
```
#include <stdio.h>
#include <string.h>
#include <stdlib.h>
#include <stdbool.h>
#define SIZE 20
struct DataItem
{
 int data;
  int key;
};
struct DataItem* hashArray[SIZE];
struct DataItem* dummyItem;
struct DataItem* item;
int hashCode(int key) {
 return key % SIZE;
}
struct DataItem *search(int key) {
 int hashIndex = hashCode(key);
 while(hashArray[hashIndex] != NULL) {
 if(hashArray[hashIndex]->key == key)
 return hashArray[hashIndex];
 ++hashIndex;
 hashIndex %= SIZE;
 }
 return NULL;
}
void insert(int key,int data) {
 struct DataItem *item = (struct DataItem*) malloc(sizeof(struct DataItem));
 item->data = data;
 item->key = key;
 int hashIndex = hashCode(key);
```

```
 while(hashArray[hashIndex] != NULL && hashArray[hashIndex]->key != -
1) {
 ++hashIndex;
 hashIndex %= SIZE;
 }
 hashArray[hashIndex] = item;
}
struct DataItem* delete(struct DataItem* item)
{
 int key = item->key;
 int hashIndex = hashCode(key);
 while(hashArray[hashIndex] != NULL) {
 if(hashArray[hashIndex]->key == key) {
 struct DataItem* temp = hashArray[hashIndex];
 hashArray[hashIndex] = dummyItem;
 return temp;
 }
 ++hashIndex;
 hashIndex %= SIZE;
 }

 return NULL;
}
void display() {
 int i = 0;
 for(i = 0; i<SIZE; i++) {
 if(hashArray[i] != NULL)
 printf(" (%d,%d)",hashArray[i]->key,hashArray[i]->data);
 else
 printf(" ~~ ");
 }
```

```
 printf("\n");
}
int main() {
 dummyItem = (struct DataItem*) malloc(sizeof(struct DataItem));
 dummyItem->data = -1;
 dummyItem->key = -1;
 insert(1, 20);
 insert(2, 70);
 insert(42, 80);
 insert(4, 25);
 insert(12, 44);
 insert(14, 32);
 insert(17, 11);
 insert(13, 78);
 insert(37, 97);
 display();
 item = search(37);
 if(item != NULL) {
 printf("Element found: %d\n", item->data);
  } else {
 printf("Element not found\n");
 }
 delete(item);
 item = search(37);
 if(item != NULL) {
 printf("Element found: %d\n", item->data);
 } else {
 printf("Element not found\n");
 }
}
```


Element found: 97
Element not found
Process returned 0 (0x0)   execution time : 0.065 s
Press any key to continue.