

CSE2001 (Data Structures & Algorithms) Lab-8

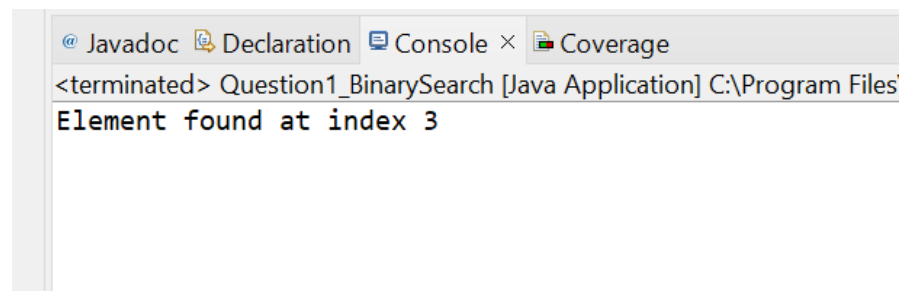
KHAN MOHD OWAIS RAZA

20BCD7138

Q1] Write a program for linear search and binary search

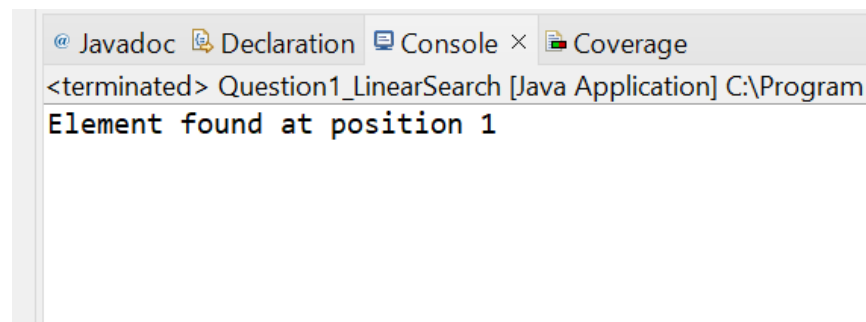
Java code for binary search :-

```
/**
Name: KHAN MOHD OWAIS RAZA
ID : 20BCD7138
Course: Data Structures & Algorithm
Code: CSE2001
Slot: L19+L20
**/
/* Lab-8 (12-11-2022)*/
/* Java code to implement binary search */
package CSE2001_Lab8_20BCD7138;
public class Question1_BinarySearch {
int binarySearch(int arr[], int l, int r, int x){
if (r >= l) {
int mid = l + (r - l) / 2;
if (arr[mid] == x) return mid;
if (arr[mid] > x) return binarySearch(arr, l, mid - 1, x);
return binarySearch(arr, mid + 1, r, x);
}
return -1;
}
public static void main(String args[]){
Question1_BinarySearch ob = new Question1_BinarySearch();
int arr[] = { 2, 3, 4, 10, 40 };
int n = arr.length;
int x = 10;
int result = ob.binarySearch(arr, 0, n - 1, x);
if (result == -1)
System.out.println("Element is not present");
else
System.out.println("Element found at index " + result);
}}
```



Java code for linear search :-

```
/**
Name: KHAN MOHD OWAIS RAZA
ID : 20BCD7138
Course: Data Structures & Algorithm
Code: CSE2001
Slot: L19+L20
**/
/* Lab-8 (12-11-2022)*/
/* Java code to implement linear search */
package CSE2001_Lab8_20BCD7138;
public class Question1_LinearSearch {
    static int search(int arr[], int n, int x){
        for (int i = 0; i < n; i++) {
            if (arr[i] == x) return i;
        }
        return -1;
    }
    public static void main(String[] args){
        int[] arr = { 3, 4, 1, 7, 5 };
        int n = arr.length;
        int x = 4;
        int index = search(arr, n, x);
        if (index == -1)
            System.out.println("Element is not present in the array");
        else
            System.out.println("Element found at position " + index);
    }
}
```



Java code for implementing binary search and linear search :-

```
/**
Name: KHAN MOHD OWAIS RAZA
ID : 20BCD7138
Course: Data Structures & Algorithm
Code: CSE2001
Slot: L19+L20
**/
/* Lab-8 (12-11-2022)*/
/* Java code to implement binary search and linear search */
package CSE2001_Lab8_20BCD7138;
import java.io.*;
import java.lang.*;
import java.util.*;
public class Question1_Binary_and_Linear_Search{
public static void main(String args[])throws IOException{
System.out.println("Java code for implementing binary search and linear
search");
System.out.println("-----
----");
BufferedReader br=new BufferedReader(new InputStreamReader(System.in));
int ch;
do{
System.out.println("\n[1] Linear search\n[2] Binary search\n[3] Binary
search with recursion\n[4] Exit");
ch=Integer.parseInt(br.readLine());
if(ch==4)return;
System.out.println("Enter the number of terms:");
int n=Integer.parseInt(br.readLine());
int a[]=new int[n];
for(int i=0;i<n;i++){
System.out.println("Enter "+(i+1)+"th term:");
a[i]=Integer.parseInt(br.readLine());
}
System.out.println("Enter number to be searched:");
int num=Integer.parseInt(br.readLine());
switch(ch){
case 1:LinearSearch(a,n,num);
break;
case 2:BinarySearch(a,n,num);
break;
case 3:int first=0;
int last=n;
BinarySearchRec(a,n,num,first,last);
break;
}}
while(ch!=4);
}
public static void LinearSearch(int a[],int n,int num){
for(int i=0;i<n;i++){
if(a[i]==num){
System.out.println(num+" found at "+(i+1)+"th position");
return;
}
```

```

}}
System.out.println(num+" not found");
}
public static void BinarySearch(int a[],int n,int num){
    int first=0;
    int last=n;
    int mid=(first+last)/2;
    while(first!=last){
        if(a[mid]==num){
            System.out.println(num+" found at "+(mid+1)+"th position");
            return;
        }
        if(a[mid]<num){
            first=mid+1;
            last=n;
            mid=(first+last)/2;
        }
        else{
            first=0;
            last=mid-1;
            mid=(first+last)/2;
        }
    }
    System.out.println(num+" not found");
}
public static void BinarySearchRec(int a[],int n,int num,int first, int
last){
    if(first>last){
        System.out.println(num+" not found");
        return;
    }
    int mid=(first+last)/2;
    try{
        if(a[mid]==num){
            System.out.println(num+" found at "+(mid+1)+"th position");
            return;
        }
        if(a[mid]<num){
            BinarySearchRec( a, n, num, mid+1, last);
        }
        if(a[mid]>num){
            BinarySearchRec( a, n, num, first, mid-1);
        }
    }
    catch(ArrayIndexOutOfBoundsException exception) {
        System.out.println(num+" not found");
    }
}
}
}

```

```
@ Javadoc Declaration Console × Coverage
<terminated> Question1_Binary_and_Linear_Search [Java Application] C:\Program F
Java code for implementing binary search and linear search
-----

[1] Linear search
[2] Binary search
[3] Binary search with recursion
[4] Exit
1
Enter the number of terms:
5
Enter 1th term:
10
Enter 2th term:
20
Enter 3th term:
30
Enter 4th term:
40
Enter 5th term:
50
Enter number to be searched:
20
20 found at 2th position

[1] Linear search
[2] Binary search
[3] Binary search with recursion
[4] Exit
2
Enter the number of terms:
5
Enter 1th term:
10
Enter 2th term:
20
Enter 3th term:
30
Enter 4th term:
40
Enter 5th term:
50
Enter number to be searched:
30
30 found at 3th position

[1] Linear search
[2] Binary search
[3] Binary search with recursion
[4] Exit
3
Enter the number of terms:
5
Enter 1th term:
10
Enter 2th term:
20
Enter 3th term:
30
Enter 4th term:
40
Enter 5th term:
50
Enter number to be searched:
10
10 found at 1th position
```

```
[1] Linear search
[2] Binary search
[3] Binary search with recursion
[4] Exit
```

4

Java code for comparing linear and binary search :-

```
/**
Name: KHAN MOHD OWAIS RAZA
ID : 20BCD7138
Course: Data Structures & Algorithm
Code: CSE2001
Slot: L19+L20
**/
/* Lab-8 (12-11-2022)*/
/* Java code to compare binary search and linear search */
package CSE2001_Lab8_20BCD7138;
import java.util.Random;
import java.util.Scanner;
public class Question1_Compare_Binary_and_Linear_Search {
    public static int N = 1000;
    public static int[] sequence = new int[N];
    public static boolean sequentialSearch(int[] sequence, int key) {
        for (int i = 0; i < sequence.length; i++)
            if (sequence[i] == key)
                return true;
        return false;
    }
    public static boolean binarySearch(int[] sequence, int key) {
        int low = 0, high = sequence.length - 1;
        while (low <= high){
            int mid = (low + high) / 2;
            if (key < sequence[mid]) high = mid - 1;
            else if (key > sequence[mid]) low = mid + 1;
            else return true;
        }
        return false;
    }
    public static void QUICK_SORT_ALGORITHM(int left, int right){
        if (right - left <= 0)
            return;
        else {
            int pivot = sequence[right];
            int partition = partitionIt(left, right, pivot);
            QUICK_SORT_ALGORITHM(left, partition - 1);
            QUICK_SORT_ALGORITHM(partition + 1, right);
        }
    }
    public static int partitionIt(int left, int right, long pivot){
        int leftPtr = left - 1;
        int rightPtr = right;
        while (true){
            while (sequence[++leftPtr] < pivot)
                ;
            while (rightPtr > 0 && sequence[--rightPtr] > pivot)
                ;
        }
    }
}
```

```

if (leftPtr >= rightPtr)
break;
else
swap(leftPtr, rightPtr);
}
swap(leftPtr, right);
return leftPtr;
}
public static void swap(int dex1, int dex2) {
int temp = sequence[dex1];
sequence[dex1] = sequence[dex2];
sequence[dex2] = temp;
}
public static void main(String args[]) {
System.out.println("Java code for comparing binary search and linear
search");
System.out.println("-----
-");
Random random = new Random();
for (int i = 0; i < N; i++)
sequence[i] = Math.abs(random.nextInt(100));
Scanner sc = new Scanner(System.in);
System.out.println("\nEnter the key to be searched: ");
int k = sc.nextInt();
System.out.println("Time taken to search key using linear search: ");
long startTime = System.nanoTime();
boolean result = sequentialSearch(sequence, k);
long endTime = System.nanoTime();
if (result == true)
System.out.println("Key found in " + (endTime - startTime)
+ " nanoseconds");
else
System.out.println("Key doesn't exist, execution time "
+ (endTime - startTime) + " nanoseconds");
System.out.println("Time taken to search key using binary search: ");
QUICK_SORT_ALGORITHM(0, N - 1);
startTime = System.nanoTime();
result = sequentialSearch(sequence, k);
endTime = System.nanoTime();
if (result == true)
System.out.println("Key found in " + (endTime - startTime)
+ " nanoseconds");
else
System.out.println("Key doesn't exist, execution time "
+ (endTime - startTime) + " nanoseconds");
sc.close();
}}

```

```

<terminated> Question1_Compare_Binary_and_Linear_Search [Java Application]
Java code for comparing binary search and linear search
-----

```

```

Enter the key to be searched:
50
Time taken to search key using linear search:
Key found in 9800 nanoseconds
Time taken to search key using binary search:
Key found in 3400 nanoseconds

```

Q2] Write a program to implement basic hashing techniques

C code for implementing basic hashing techniques :-

```
/**
Name: KHAN MOHD OWAIS RAZA
ID : 20BCD7138
Course: Data Structures & Algorithm
Code: CSE2001
Slot: L19+L20
**/
/* Lab-8 (12-11-2022)*/
/* C code to implement basic hashing techniques */
#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(" - \n");
    }
    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");
    }
}

```

```

1  /**
2  Name: KHAN MOHD OWAIS RAZA
3  ID : 20BCD7138
4  Course: Data Structures & Algorithm
5  Code: CSE2001
6  Slot: L19+L20
7  **/
8  /* Lab-8 (12-11-2022)*/
9  /* C code to implement basic hashing techniques */
10 #include <stdio.h>
11 #include <string.h>
12 #include <stdlib.h>
13 #include <stdbool.h>
14 #define SIZE 20
15 struct DataItem {
16     int data;
17     int key;
18 };
19 struct DataItem* hashArray[SIZE];
20 struct DataItem* dummyItem;
21 struct DataItem* item;
22 int hashCode(int key) {
23     return key % SIZE;
24 }
25 struct DataItem *search(int key) {
26     int hashIndex = hashCode(key);
27     while(hashArray[hashIndex] != NULL) {
28         if(hashArray[hashIndex]->key == key)
29             return hashArray[hashIndex];
30         ++hashIndex;
31         hashIndex %= SIZE;
32     }
33     return NULL;
34 }
35 void insert(int key,int data) {
36     struct DataItem *item = (struct DataItem*) malloc(sizeof(struct DataItem));
37     item->data = data;
38     item->key = key;
39     int hashIndex = hashCode(key);
40     while(hashArray[hashIndex] != NULL && hashArray[hashIndex]->key != -1) {
41         ++hashIndex;
42         hashIndex %= SIZE;
43     }
44     hashArray[hashIndex] = item;
45 }
46 struct DataItem* delete(struct DataItem* item) {
47     int key = item->key;
48     int hashIndex = hashCode(key);
49     while(hashArray[hashIndex] != NULL) {
50         if(hashArray[hashIndex]->key == key) {
51             struct DataItem* temp = hashArray[hashIndex];
52             hashArray[hashIndex] = dummyItem;
53             return temp;
54         }
55         ++hashIndex;
56         hashIndex %= SIZE;
57     }
58     return NULL;
59 }
60 void display() {
61     int i = 0;
62     for(i = 0; i<SIZE; i++) {
63         if(hashArray[i] != NULL)
64             printf(" (%d,%d)",hashArray[i]->key,hashArray[i]->data);
65         else
66             printf(" - \n");
67     }
68     printf("\n");
69 }

```

```

70 ☐ int main() {
71     dummyItem = (struct DataItem*) malloc(sizeof(struct DataItem));
72     dummyItem->data = -1;
73     dummyItem->key = -1;
74     insert(1, 20);
75     insert(2, 70);
76     insert(42, 80);
77     insert(4, 25);
78     insert(12, 44);
79     insert(14, 32);
80     insert(17, 11);
81     insert(13, 78);
82     insert(37, 97);
83     display();
84     item = search(37);
85 ☐ if(item != NULL) {
86     printf("Element found: %d\n", item->data);
87     } else {
88     printf("Element not found\n");
89     }
90     delete(item);
91     item = search(37);
92 ☐ if(item != NULL) {
93     printf("Element found: %d\n", item->data);
94     } else {
95     printf("Element not found\n");
96     }}
97
98

```

```

C:\Users\Owais\Desktop\Question2.exe
-
(1,20) (2,70) (42,80) (4,25) -
-
-
-
-
-
-
(12,44) (13,78) (14,32) -
-
(17,11) (37,97) -

Element found: 97
Element not found

-----
Process exited after 0.04723 seconds with return value 0
Press any key to continue . . .

```

Q4] Write a program to implement binary search for given sequence 12, 33, 42, 51, 66, 73, 87, 99, 101.

C code for the question :-

```
/**
Name: KHAN MOHD OWAIS RAZA
ID : 20BCD7138
Course: Data Structures & Algorithm
Code: CSE2001
Slot: L19+L20
**/
/* Lab-8 (12-11-2022)*/
/* Java code to implement binary search for given sequence */
#include <stdio.h>
void binary_search(int array[], int first, int last, int n){
    int i ,middle;
    middle = (first + last) / 2;
    while (first <= last) {
        if (array[middle] < n)
            first = middle + 1;
        else if (array[middle] == n) {
            printf("%d found at location %d.\n", n, middle+1);
            break;
        }
        else
            last = middle - 1;
        middle = (first + last) / 2;
    }
    if ( first > last )
        printf("Not found! %d is not present in the list.\n", n);
}
search(int arr[], int size, int data){
    int p = (size - 1) / 2, low, high, a1 = 0, a2 = 1, i = 1;
    low = p + a1;
    high = p + a2;
    while(i){
        if(data >= arr[low] && data <= arr[high]){
            binary_search(arr, low, high, data);
            break;
        }
        else if(data < arr[low]){
            binary_search(arr, 0, low, data);
            break;
        }
        else{
            a2 = a2 * 2;
            low = high;
            high = p + a2;
        }
    }
}
int main(){
```

```

int a[200], i, j, n, size;
printf("Enter the size of the list:");
scanf("%d", &size);
printf("Enter %d Integers in ascending order\n", size);
for (i = 0; i < size; i++)
scanf("%d", &a[i]);
printf("Enter value to find\n");
scanf("%d", &n);
search(a, size, n);
return 0;
}

```

Question4.c

```

1  #include <stdio.h>
2  void binary_search(int array[], int first, int last, int n){
3      int i ,middle;
4      middle = (first + last) / 2;
5      while (first <= last) {
6          if (array[middle] < n)
7              first = middle + 1;
8          else if (array[middle] == n) {
9              printf("%d found at location %d.\n", n, middle+1);
10             break;
11         }
12         else
13             last = middle - 1;
14         middle = (first + last) / 2;
15     }
16     if ( first > last )
17         printf("Not found! %d is not present in the list.\n", n);
18 }
19 search(int arr[], int size, int data){
20     int p = (size - 1) / 2, low, high, a1 = 0, a2 = 1, i = 1;
21     low = p + a1;
22     high = p + a2;
23     while(i){
24         if(data >= arr[low] && data <= arr[high]){
25             binary_search(arr, low, high, data);
26             break;
27         }
28         else if(data < arr[low]){
29             binary_search(arr, 0, low, data);
30             break;
31         }
32         else{
33             a2 = a2 * 2;
34             low = high;
35             high = p + a2;
36         }
37     }
38     int main(){
39         int a[200], i, j, n, size;
40         printf("Enter the size of the list:");
41         scanf("%d", &size);
42         printf("Enter %d Integers in ascending order\n", size);
43         for (i = 0; i < size; i++)
44             scanf("%d", &a[i]);
45         printf("Enter value to find\n");
46         scanf("%d", &n);
47         search(a, size, n );
48         return 0;
49     }
50

```

```
C:\Users\Owais\Desktop\Question4.exe
Enter 9 Integers in ascending order
12
33
42
51
66
73
87
99
101
Enter value to find
51
51 found at location 4.

-----
Process exited after 70.39 seconds with return value 0
Press any key to continue . . .
```

Q5] Write a program to illustrate the concept of hashing and resolve collision if there is any.

C code to illustrate concept of hashing :-

```
/**
Name: KHAN MOHD OWAIS RAZA
ID : 20BCD7138
Course: Data Structures & Algorithm
Code: CSE2001
Slot: L19+L20
**/
/* Lab-8 (12-11-2022)*/
/* C code to illustrate concept of hashing */
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#define CAPACITY 500
unsigned long hash_function(char* str) {
    unsigned long i = 0;
    for (int j=0; str[j]; j++)
        i += str[j];
    return i % CAPACITY;
}
typedef struct Ht_item Ht_item;
struct Ht_item {
    char* key;
    char* value;
};
typedef struct HashTable HashTable;
```

```

struct HashTable {
    Ht_item** items;
    int size;
    int count;
};

Ht_item* create_item(char* key, char* value) {
    Ht_item* item = (Ht_item*) malloc (sizeof(Ht_item));
    item->key = (char*) malloc (strlen(key) + 1);
    item->value = (char*) malloc (strlen(value) + 1);
    strcpy(item->key, key);
    strcpy(item->value, value);
    return item;
}

HashTable* create_table(int size) {
    HashTable* table = (HashTable*) malloc (sizeof(HashTable));
    table->size = size;
    table->count = 0;
    table->items = (Ht_item**) calloc (table->size, sizeof(Ht_item*));
    for (int i=0; i<table->size; i++)
        table->items[i] = NULL;
    return table;
}

void free_item(Ht_item* item) {
    free(item->key);
    free(item->value);
    free(item);
}

void free_table(HashTable* table) {
    for (int i=0; i<table->size; i++) {
        Ht_item* item = table->items[i];
        if (item != NULL)
            free_item(item);
    }
    free(table->items);
    free(table);
}

void handle_collision(HashTable* table, unsigned long index, Ht_item* item) {
}

void ht_insert(HashTable* table, char* key, char* value) {
    Ht_item* item = create_item(key, value);
    unsigned long index = hash_function(key);
    Ht_item* current_item = table->items[index];
    if (current_item == NULL) {
        if (table->count == table->size) {
            printf("Insert Error: Hash Table is full\n");
            free_item(item);
            return;
        }
        table->items[index] = item;
        table->count++;
    }
    else {

```

```

if (strcmp(current_item->key, key) == 0) {
    strcpy(table->items[index]->value, value);
    return;
}
else {
    handle_collision(table, index, item);
    return;
}
}
}
char* ht_search(HashTable* table, char* key) {
    int index = hash_function(key);
    Ht_item* item = table->items[index];
    if (item != NULL) {
        if (strcmp(item->key, key) == 0)
            return item->value;
    }
    return NULL;
}
void print_search(HashTable* table, char* key) {
    char* val;
    if ((val = ht_search(table, key)) == NULL) {
        printf("Key:%s does not exist\n", key);
        return;
    }
    else {
        printf("Key:%s, Value:%s\n", key, val);
    }
}
void print_table(HashTable* table) {
    printf("\nHash Table\n-----\n");
    for (int i=0; i<table->size; i++) {
        if (table->items[i]) {
            printf("Index:%d, Key:%s, Value:%s\n", i, table->items[i]->key, table->items[i]->value);
        }
    }
    printf("-----\n\n");
}
int main() {
    HashTable* ht = create_table(CAPACITY);
    ht_insert(ht, "1", "First address");
    ht_insert(ht, "2", "Second address");
    print_search(ht, "1");
    print_search(ht, "2");
    print_search(ht, "3");
    print_table(ht);
    free_table(ht);
    return 0;
}

```



```

1  /**
2  Name: KHAN MOHD OWAIS RAZA
3  ID : 20BCD7138
4  Course: Data Structures & Algorithm
5  Code: CSE2001
6  Slot: L19+L20
7  **/
8  /* Lab-8 (12-11-2022)*/
9  /* Java code to illustrate concept of hashing */
10 #include <stdio.h>
11 #include <stdlib.h>
12 #include <string.h>
13 #define CAPACITY 500
14 unsigned long hash_function(char* str) {
15     unsigned long i = 0;
16     for (int j=0; str[j]; j++)
17         i += str[j];
18     return i % CAPACITY;
19 }
20 typedef struct Ht_item Ht_item;
21 struct Ht_item {
22     char* key;
23     char* value;
24 };
25 typedef struct HashTable HashTable;
26 struct HashTable {
27     Ht_item** items;
28     int size;
29     int count;
30 };
31 Ht_item* create_item(char* key, char* value) {
32     Ht_item* item = (Ht_item*) malloc (sizeof(Ht_item));
33     item->key = (char*) malloc (strlen(key) + 1);
34     item->value = (char*) malloc (strlen(value) + 1);
35     strcpy(item->key, key);
36     strcpy(item->value, value);
37     return item;
38 }
39 HashTable* create_table(int size) {
40     HashTable* table = (HashTable*)
41     malloc (sizeof(HashTable));
42     table->size = size;
43     table->count = 0;
44     table->items = (Ht_item**)
45     calloc (table->size, sizeof(Ht_item));
46     for (int i=0; i<table->size; i++)
47         table->items[i] = NULL;
48     return table;
49 }
50 void free_item(Ht_item* item) {
51     free(item->key);
52     free(item->value);
53     free(item);
54 }

```

```

55 void free_table(HashTable* table) {
56     for (int i=0; i<table->size; i++) {
57         Ht_item* item = table->items[i];
58         if (item != NULL)
59             free_item(item);
60     }
61     free(table->items);
62     free(table);
63 }
64 void handle_collision(HashTable* table,
65 unsigned long index, Ht_item* item) {
66 }
67 void ht_insert(HashTable* table,
68 char* key, char* value) {
69     Ht_item* item = create_item(key, value);
70     unsigned long index = hash_function(key);
71     Ht_item* current_item = table->items[index];
72     if (current_item == NULL) {
73         if (table->count == table->size) {
74             printf("Insert Error: Hash Table is full\n");
75             free_item(item);
76             return;
77         }
78         table->items[index] = item;
79         table->count++;
80     }
81     else {
82         if (strcmp(current_item->key, key) == 0) {
83             strcpy(table->items[index]->value, value);
84             return;
85         }
86         else {
87             handle_collision(table, index, item);
88             return;
89         }
90     }
91     char* ht_search(HashTable* table, char* key) {
92         int index = hash_function(key);
93         Ht_item* item = table->items[index];
94         if (item != NULL) {
95             if (strcmp(item->key, key) == 0)
96                 return item->value;
97             return NULL;
98         }
99     }
100 void print_search(HashTable* table, char* key) {
101     char* val;
102     if ((val = ht_search(table, key)) == NULL) {
103         printf("Key:%s does not exist\n", key);
104         return;
105     }
106     else {
107         printf("Key:%s, Value:%s\n", key, val);
108     }
109 }

```

```

108 void print_table(HashTable* table) {
109     printf("\nHash Table\n-----\n");
110     for (int i=0; i<table->size; i++) {
111         if (table->items[i]) {
112             printf("Index:%d, Key:%s, Value:%s\n", i,
113                 table->items[i]->key, table->items[i]->value);
114         }
115     }
116     printf("-----\n\n");
117 }
118 int main() {
119     HashTable* ht = create_table(CAPACITY);
120     ht_insert(ht, "1", "First address");
121     ht_insert(ht, "2", "Second address");
122     print_search(ht, "1");
123     print_search(ht, "2");
124     print_search(ht, "3");
125     print_table(ht);
126     free_table(ht);
127     return 0;
128 }
129

```

C:\Users\Owais\Desktop\Question5.exe

```

Key:1, Value:First address
Key:2, Value:Second address
Key:3 does not exist

```

Hash Table

```

-----
Index:49, Key:1, Value:First address
Index:50, Key:2, Value:Second address
-----

```

```

Process returned 0 (0x0)   execution time : 0.936 s
Press any key to continue.

```

C code for resolving collision in hashing :-

```
/**
Name: KHAN MOHD OWAIS RAZA
ID : 20BCD7138
Course: Data Structures & Algorithm
Code: CSE2001
Slot: L19+L20
**/
/* Lab-8 (12-11-2022)*/
/* C code to avoid collision in hashing */
#include <stdio.h>
#include <stdlib.h>
#define TABLE_SIZE 10
struct node{
    int data;
    struct node *next;
};
struct node *head[TABLE_SIZE]={NULL},*c;
void insert(){
    int i,key;
    printf("\nEnter a value to insert into hash table:\n");
    scanf("%d",&key);
    i=key%TABLE_SIZE;
    struct node * newnode=(struct node *)
    malloc(sizeof(struct node));
    newnode->data=key;
    newnode->next = NULL;
    if(head[i] == NULL)
        head[i] = newnode;
    else{
        c=head[i];
        while(c->next != NULL){
            c=c->next;
        }
        c->next=newnode;
    }
}
void search()
{
    int key,index;
    printf("\nEnter the element to be searched:\n");
    scanf("%d",&key);
    index=key%TABLE_SIZE;
    if(head[index] == NULL)
        printf("\nElement is not present\n");
    else{
        for(c=head[index];c!=NULL;c=c->next){
            if(c->data == key){
                printf("Element is present\n");
                break;
            }
        }
        if(c==NULL)
```



```

printf("\nElement is not present\n");
}}
void display(){
int i;
for(i=0;i<TABLE_SIZE;i++){
printf("\nEntries at index %d:\n",i);
if(head[i] == NULL){
printf("No Hash Entry \n");
}
else{
for(c=head[i];c!=NULL;c=c->next)
printf("%d ",c->data);
printf("\n");
}}}
main(){
printf("C code to avoid collision in hashing \n");
printf("----- \n");
int opt,key,i;
while(1){
printf("\nSelect an operation:");
printf("\n[1] Insert [2] Display [3] Search [4] Exit \n");
scanf("%d",&opt);
switch(opt){
case 1:insert();
break;
case 2:display();
break;
case 3:search();
break;
case 4:exit(0);
}}}

```

Question5_Collision_Resolving.c

```

1  /**
2   Name: KHAN MOHD OWAIS RAZA
3   ID : 20BCD7138
4   Course: Data Structures & Algorithm
5   Code: CSE2001
6   Slot: L19+L20
7   **/
8   /* Lab-8 (12-11-2022)*/
9   /* C code to avoid collision in hashing */
10  #include <stdio.h>
11  #include <stdlib.h>
12  #define TABLE_SIZE 10
13  struct node{
14      int data;
15      struct node *next;
16  };
17  struct node *head[TABLE_SIZE]={NULL},*c;
18  void insert(){
19      int i,key;
20      printf("\nEnter a value to insert into hash table:\n");
21      scanf("%d",&key);

```

```

22 i=key%TABLE_SIZE;
23 struct node * newnode=(struct node *)
24 malloc(sizeof(struct node));
25 newnode->data=key;
26 newnode->next = NULL;
27 if(head[i] == NULL)
28 head[i] = newnode;
29 else{
30 c=head[i];
31 while(c->next != NULL){
32 c=c->next;
33 }
34 c->next=newnode;
35 }
36 void search()
37 {
38 int key,index;
39 printf("\nEnter the element to be searched:\n");
40 scanf("%d",&key);
41 index=key%TABLE_SIZE;
42 if(head[index] == NULL)
43 printf("\nElement is not present\n");
44 else{
45 for(c=head[index];c!=NULL;c=c->next){
46 if(c->data == key){
47 printf("Element is present\n");
48 break;
49 }
50 if(c==NULL)
51 printf("\nElement is not present\n");
52 }
53 void display(){
54 int i;
55 for(i=0;i<TABLE_SIZE;i++){
56 printf("\nEntries at index %d:\n",i);
57 if(head[i] == NULL){
58 printf("No Hash Entry \n");
59 }
60 else{
61 for(c=head[i];c!=NULL;c=c->next)
62 printf("%d ",c->data);
63 printf("\n");
64 }
65 }
66 main(){
67 printf("C code to avoid collision in hashing \n");
68 printf("----- \n");
69 int opt,key,i;
70 while(1){
71 printf("\nSelect an operation:");
72 printf("\n[1] Insert [2] Display [3] Search [4] Exit \n");
73 scanf("%d",&opt);
74 switch(opt){
75 case 1:insert();
76 break;
77 case 2:display();
78 break;

```

```
78     case 3:search();
79     break;
80     case 4:exit(0);
81     }}}
82
```

```
C:\Users\Owais\Desktop\Question5_Collision_Reso...
C code to avoid collision in hashing
-----

Select an operation:
[1] Insert [2] Display [3] Search [4] Exit
1

Enter a value to insert into hash table:
10

Select an operation:
[1] Insert [2] Display [3] Search [4] Exit
1

Enter a value to insert into hash table:
20

Select an operation:
[1] Insert [2] Display [3] Search [4] Exit
1

Enter a value to insert into hash table:
30

Select an operation:
[1] Insert [2] Display [3] Search [4] Exit
1

Enter a value to insert into hash table:
40

Select an operation:
[1] Insert [2] Display [3] Search [4] Exit
1

Enter a value to insert into hash table:
50

Select an operation:
[1] Insert [2] Display [3] Search [4] Exit
2

Entries at index 0:
10 20 30 40 50

Entries at index 1:
No Hash Entry

Entries at index 2:
No Hash Entry

Entries at index 3:
No Hash Entry
```

```
Entries at index 4:
No Hash Entry

Entries at index 5:
No Hash Entry

Entries at index 6:
No Hash Entry

Entries at index 7:
No Hash Entry

Entries at index 8:
No Hash Entry

Entries at index 9:
No Hash Entry

Select an operation:
[1] Insert [2] Display [3] Search [4] Exit
3

Enter the element to be searched:
20
Element is present

Select an operation:
[1] Insert [2] Display [3] Search [4] Exit
3

Enter the element to be searched:
35

Element is not present

Select an operation:
[1] Insert [2] Display [3] Search [4] Exit
4

-----
Process exited after 38.94 seconds with return value 0
Press any key to continue . . .
```


Q3] Write a program that takes person's details (name, age, city) and search specific person name using linear search.

```
/**
Name: KHAN MOHD OWAIS RAZA
ID : 20BCD7138
Course: Data Structures & Algorithm
Code: CSE2001
Slot: L19+L20
**/
/* Lab-8 (12-11-2022)*/
/* C code to search specific person using linear search */
#include<stdio.h>
int main(){
int ID;
printf("Enter ID to search: ");
scanf("%d",&ID);
switch(ID) {
case 1:
printf("NAME: Cristiano Ronaldo \nAGE: 37 \nCITY: Funchal, Portugal");
break;
case 2:
printf("NAME: Lionel Messi \nAGE: 35 \nCITY: Rosario, Argentina");
break;
case 3:
printf("NAME: Zinedine Zidane \nAGE: 50 \nCITY: Marseille, France");
break;
case 4:
printf("NAME: Neymar da Silva \nAGE: 30 \nCITY: São Paulo, Brazil");
break;
case 5:
printf("NAME: Rocky Balboa \nAGE: 30 \nCITY: Philadelphia, United States");
break;
default:
printf("Please provide valid ID!");
}
return 0;
}
```

Question3.c

```
1  /**
2  Name: KHAN MOHD OWAIS RAZA
3  ID : 20BCD7138
4  Course: Data Structures & Algorithm
5  Code: CSE2001
6  Slot: L19+L20
7  */
8  /* Lab-8 (12-11-2022)*/
9  /* C code to search specific person using linear search */
10 #include<stdio.h>
11 int main(){
12     int ID;
13     printf("Enter ID to search: ");
14     scanf("%d",&ID);
15     switch(ID) {
16     case 1:
17         printf("NAME: Cristiano Ronaldo \nAGE: 37 \nCITY: Funchal, Portugal");
18         break;
19     case 2:
20         printf("NAME: Lionel Messi \nAGE: 35 \nCITY: Rosario, Argentina");
21         break;
22     case 3:
23         printf("NAME: Zinedine Zidane \nAGE: 50 \nCITY: Marseille, France");
24         break;
25     case 4:
26         printf("NAME: Neymar da Silva \nAGE: 30 \nCITY: São Paulo, Brazil");
27         break;
28     case 5:
29         printf("NAME: Rocky Balboa \nAGE: 30 \nCITY: Philadelphia, United States");
30         break;
31     default:
32         printf("Please provide valid ID!");
33     }
34     return 0;
35 }
36
```

C:\Users\Owais\Desktop\Question3.exe

```
Enter ID to search: 1
NAME: Cristiano Ronaldo
AGE: 37
CITY: Funchal, Portugal
-----
Process exited after 8.912 seconds with return value 0
Press any key to continue . . .
```

C:\Users\Owais\Desktop\Question3.exe

```
Enter ID to search: 2
NAME: Lionel Messi
AGE: 35
CITY: Rosario, Argentina
-----
Process exited after 2.018 seconds with return value 0
Press any key to continue . . .
```

C:\Users\Owais\Desktop\Question3.exe

Enter ID to search: 3

NAME: Zinedine Zidane

AGE: 50

CITY: Marseille, France

Process exited after 1.639 seconds with return value 0

Press any key to continue . . .

C:\Users\Owais\Desktop\Question3.exe

Enter ID to search: 4

NAME: Neymar da Silva

AGE: 30

CITY: São Paulo, Brazil

Process exited after 3.853 seconds with return value 0

Press any key to continue . . .

C:\Users\Owais\Desktop\Question3.exe

Enter ID to search: 5

NAME: Rocky Balboa

AGE: 30

CITY: Philadelphia, United States

Process exited after 1.902 seconds with return value 0

Press any key to continue . . .