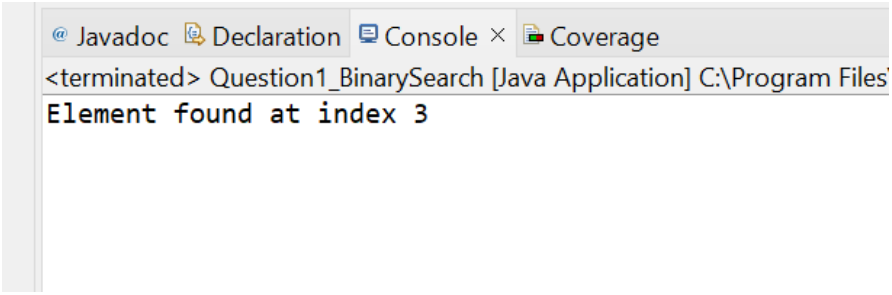
CSE2001 (Data Structures & Algorithms) Lab-8

KHAN MOHD OWAIS RAZA
20BCD7138

Q1] Write a program for linear search and binary search

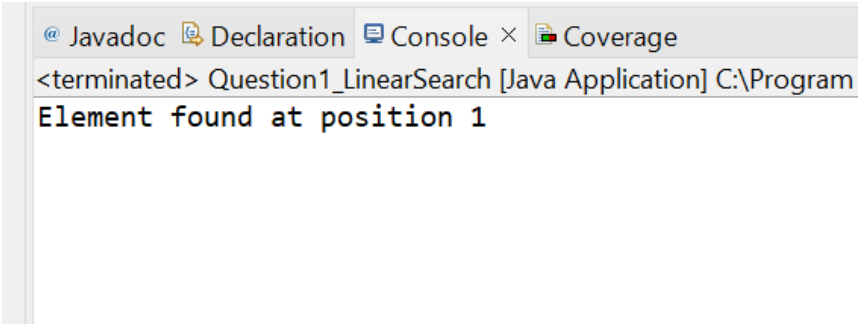
Java code for binary search :-

```
/**
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 :-

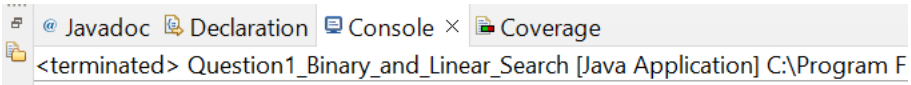
```
/**
Name: KHAN MOHD OWAIS RAZA
ID : 20BCD7138
Course: Data Structures & Algorithm
Code: CSE2001
Slot: L19+L20
**/
/* Lab-8 (12-11-2022)*/
/* Java code to implement linear search */
package CSE2001_Lab8_20BCD7138;
public class Question1_LinearSearch {
static int search(int arr[], int n, int x){
for (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");
}}}
```




```
<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 :-

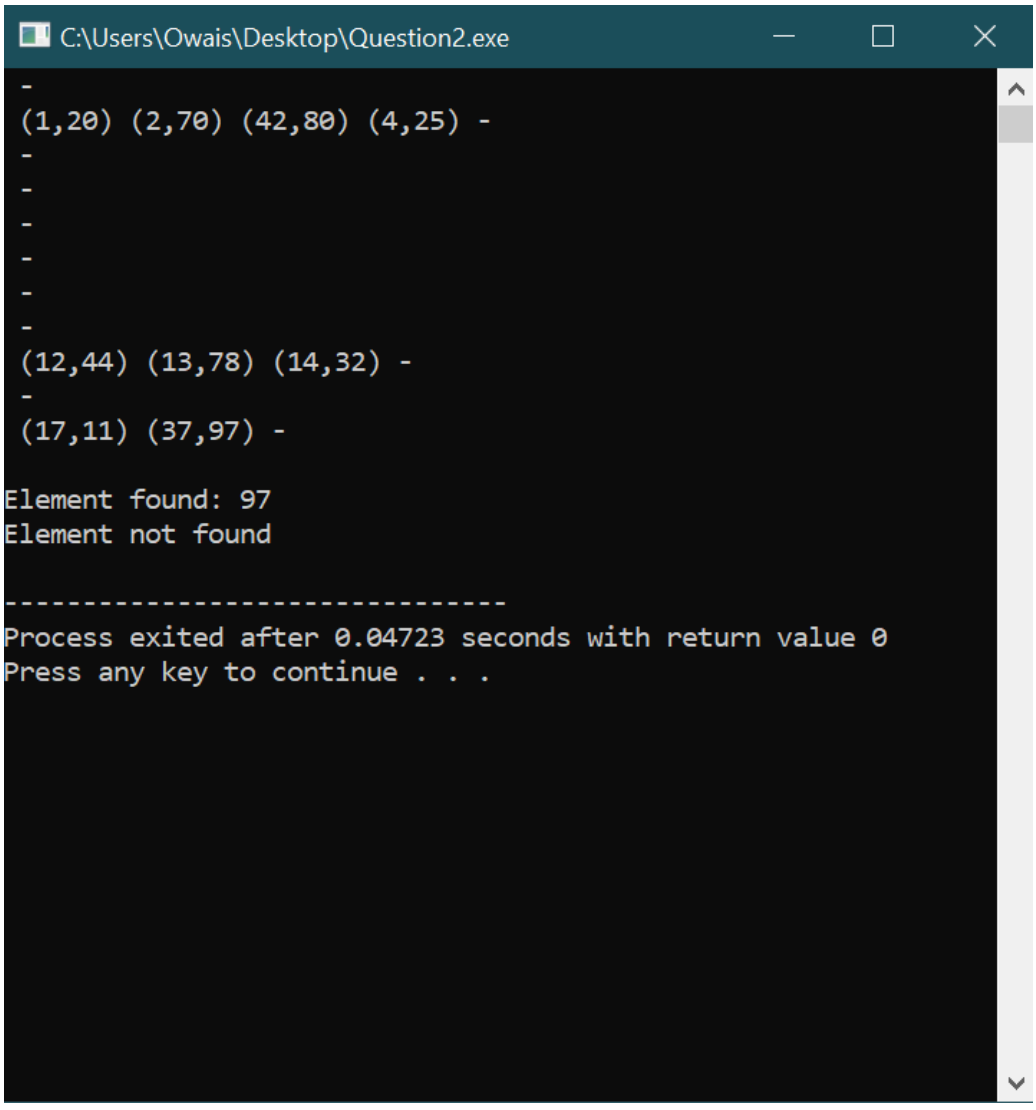
```
/**
Name: KHAN MOHD OWAIS RAZA
ID : 20BCD7138
Course: Data Structures & Algorithm
Code: CSE2001
Slot: L19+L20
**/
/* Lab-8 (12-11-2022)*/
/* C code to implement basic hashing techniques */
#include <stdio.h>
#include <string.h>
#include <stdlib.h>
#include <stdbool.h>
#define SIZE 20
struct DataItem {
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");
}}
```

Question2.c

```c
/**
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");
}
```

```c
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");
}}

```



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

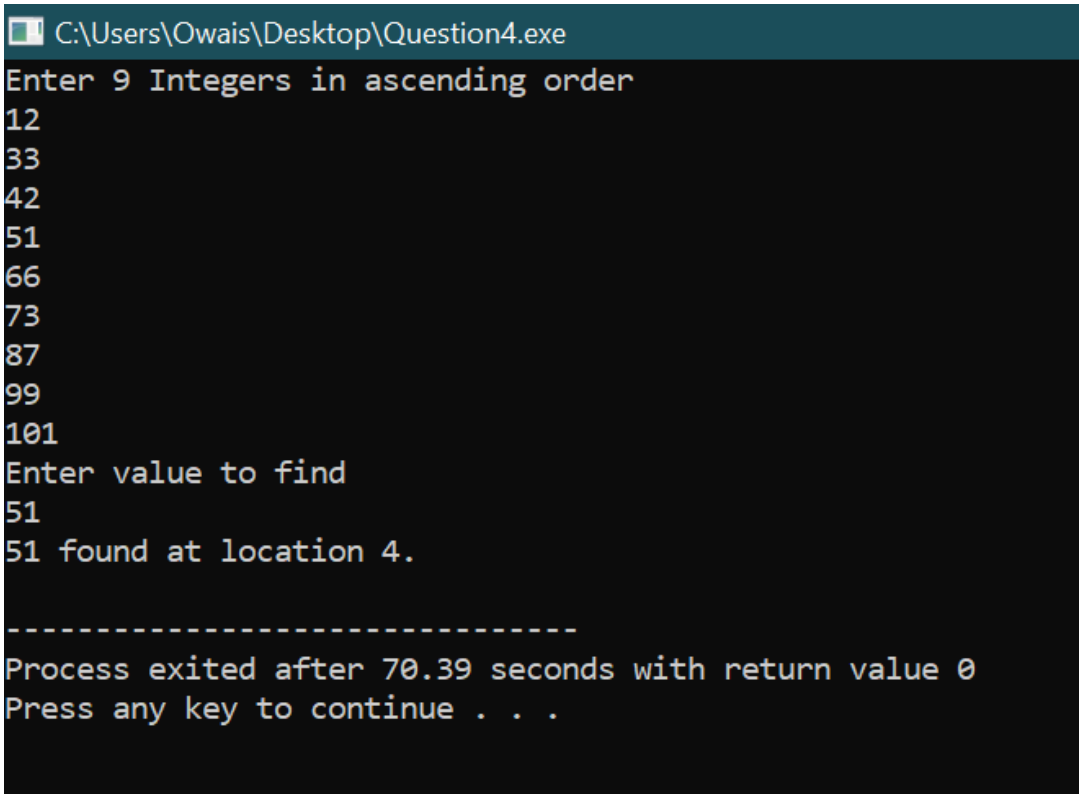
C code for the question :-

```
/**
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(){
```

```c
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

```c
#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;
}

```



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;
}
```

Question5.c

```c
/**
Name: KHAN MOHD OWAIS RAZA
ID : 20BCD7138
Course: Data Structures & Algorithm
Code: CSE2001
Slot: L19+L20
**/
/* Lab-8 (12-11-2022)*/
/* Java 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);
}
```

```c
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);
}}
```

```c
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;
}
```

```
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)
```

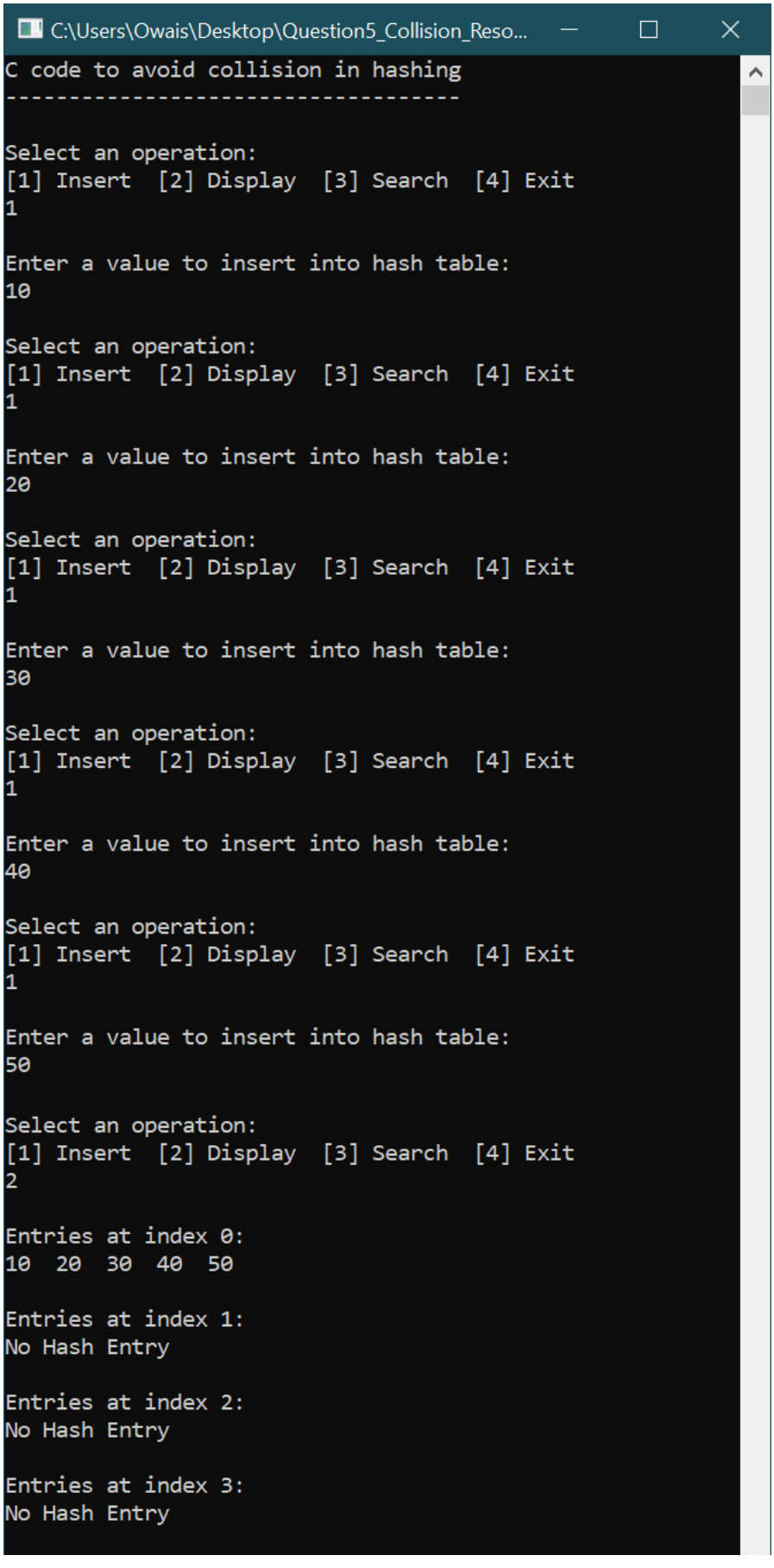
```
printf("\nElement is not present\n");
}}
void display(){
int i;
for(i=0;i<TABLE_SIZE;i++){
printf("\nEntries at index %d:\n",i);
if(head[i] == NULL){
printf("No Hash Entry \n");
}
else{
for(c=head[i];c!=NULL;c=c->next)
printf("%d  ",c->data);
printf("\n");
}}}
main(){
printf("C code to avoid collision 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);
```

```c
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;
```

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



```
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.

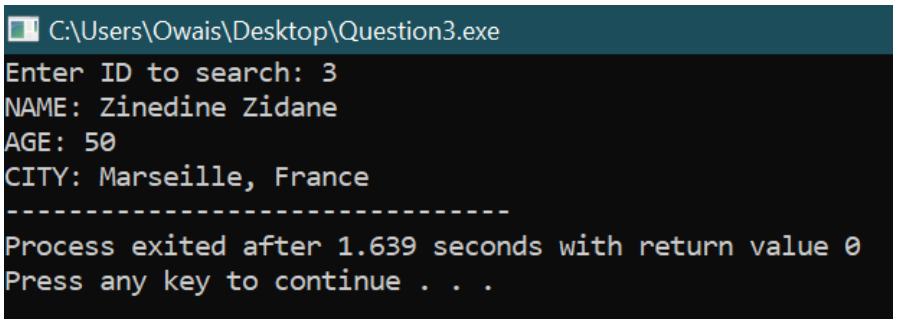
```
/**
Name: KHAN MOHD OWAIS RAZA
ID : 20BCD7138
Course: Data Structures & Algorithm
Code: CSE2001
Slot: L19+L20
**/
/* Lab-8 (12-11-2022)*/
/* C code to search specific person using linear search */
#include<stdio.h>
int main(){
int ID;
printf("Enter ID to search: ");
scanf("%d",&ID);
switch(ID) {
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

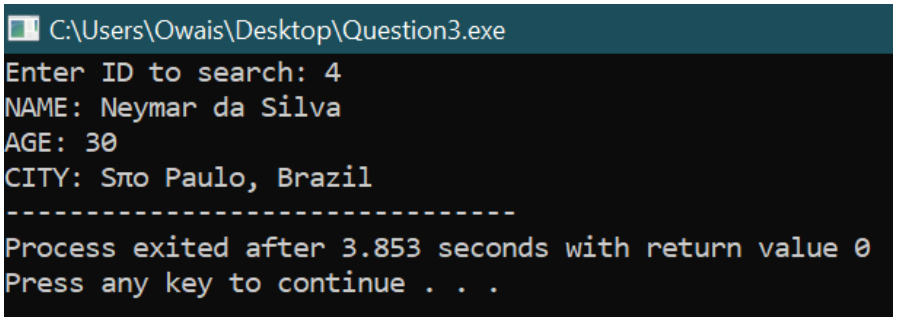
```c
/**
Name: KHAN MOHD OWAIS RAZA
ID : 20BCD7138
Course: Data Structures & Algorithm
Code: CSE2001
Slot: L19+L20
**/
/* Lab-8 (12-11-2022)*/
/* C code to search specific person using linear search */
#include<stdio.h>
int main(){
int ID;
printf("Enter ID to search: ");
scanf("%d",&ID);
switch(ID) {
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;
}

```

```
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 . . .
```

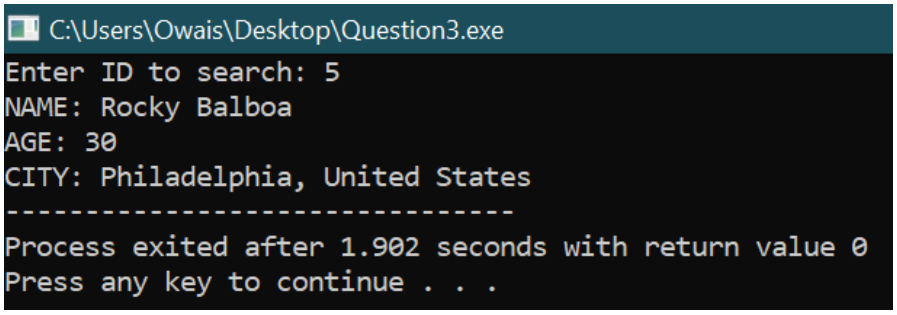
```
C:\Users\Owais\Desktop\Question3.exe
Enter ID to search: 2
NAME: Lionel Messi
AGE: 35
CITY: Rosario, Argentina
--------------------------------
Process exited after 2.018 seconds with return value 0
Press any key to continue . . .
```


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: Sπo 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 . . .