1. **Activity: Store n numbers in an unsorted array. Using a menu driven logic perform search (linear and Binary) for a given number and report success or failure in the form of suitable message.**

**NOTE: Linear search returns position in unsorted array & binary search returns position in sorted array**

**Algorithm:**

1. Linear Search:
   1. For I=0; I < size; i++
      1. If arr[i] == element
         1. Return position
   2. Return -1
2. Binary Search:
   1. Bubble Sort:
      1. int i, j, swap = 1;
      2. while (swap)
         1. for(i = 0; i < size-1; i++){
            1. swap = 0;
            2. for(j = 0; j < (size-1)-i; j++){

printf("Comparing %d %d\n", arr[j], arr[j+1]);

if (arr[j] > arr[j+1])

int t = arr[j];

arr[j] = arr[j+1];

arr[j+1] = t;

swap++;

* 1. Driver Function binary(int arr[], int l, int r, int x)
     1. if (r >= l)
        1. mid = l + (r - l) / 2;
        2. print "Mid: arr[mid]”
        3. if (arr[mid] == x)
           1. return mid;
        4. if (arr[mid] > x)
           1. return binary(arr, l, mid - 1, x);
        5. return binary(arr, mid + 1, r, x);
     2. return -1

**Program:**

#include <stdio.h>

#include <stdlib.h>

#include <time.h>

#include <ctype.h>

int size;

int linear(int arr[], int d){

    for (int i = 0; i < size; i++)

    {

        printf("Checking %d and %d.\n", arr[i], d);

        if (arr[i] == d)

        {

            return i+1;

        }

    }

    return -1;

}

int sort(int arr[]){

    int i, j, swap = 1;

    while (swap)

    {

        for(i = 0; i < size-1; i++){

            swap = 0;

            for(j = 0; j < (size-1)-i; j++){

                printf("Comparing %d %d\n", arr[j], arr[j+1]);

                if (arr[j] > arr[j+1])

                {

                    int t = arr[j];

                    arr[j] = arr[j+1];

                    arr[j+1] = t;

                    swap++;

                }

            }

        }

    }

}

int binary(int arr[], int l, int r, int x)

{

    if (r >= l) {

        int mid = l + (r - l) / 2;

        printf("Mid: %d\n", arr[mid]);

        if (arr[mid] == x)

            return mid;

        if (arr[mid] > x)

            return binary(arr, l, mid - 1, x);

        return binary(arr, mid + 1, r, x);

    }

    return -1;

}

int main(){

    srand(time(0));

    printf("Enter array size: ");

    scanf("%d", &size);

    int arr[size], sort\_arr[size];

    for (int i = 0; i < size; i++)

    {

        arr[i] = rand() % 100;

        sort\_arr[i] = arr[i];

    }

    sort(sort\_arr);

    menu: system("cls");

    printf("-------------- Array Search --------------\n");

    printf("Array:  ");

    for (int i = 0; i < size; i++)

    {

        printf("%d ", arr[i]);

    }

    printf("\n");

    printf("Sorted array:  ");

    for (int i = 0; i < size; i++)

    {

        printf("%d ", sort\_arr[i]);

    }

    printf("\n");

    printf("1. Linear Search\n");

    printf("2. Binary Search\n");

    printf("3. Exit\n");

    int ch, index, data;

    printf("Enter choice: ");

    scanf("%d", &ch);

    switch (ch)

    {

    case 1:

        printf("Enter data to search: ");

        scanf("%d", &data);

        index = linear(arr, data);

        if(index > 0) printf("Element is present in array at position: %d", index);

        else printf("Data not found");

        printf("\nPress enter to continue");

        while((getchar()) != '\n');

        getchar();

        goto menu;

        break;

    case 2:

        printf("Enter data to search: ");

        scanf("%d", &data);

        index = binary(sort\_arr, 0, size, data);

        (index == -1) ? printf("Element is not present in array")

                      : printf("Element is present in sorted array at position: %d", index+1);

        printf("\nPress enter to continue..");

        while((getchar()) != '\n');

        getchar();

        goto menu;

        break;

    case 3:

        break;

    default:

        printf("Invalid selection. Press enter to continue..");

        while((getchar()) != '\n');

        getchar();

        goto menu;

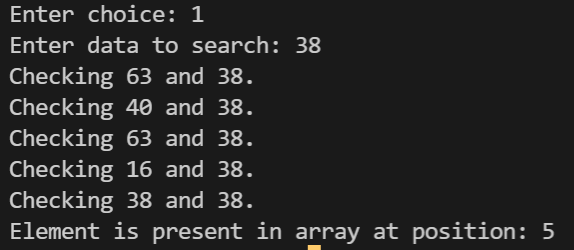
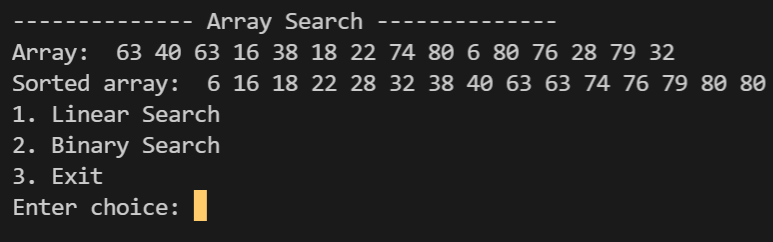
        break;

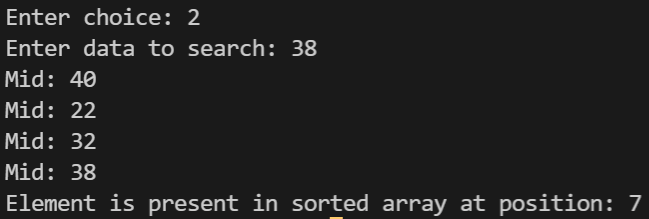
    }

    return 0;

}

**Screenshot of compilation and execution:**

****

****

1. **Activity: Store n numbers in a Linked Lists. Perform Linear search for a given number and report success or failure in the form of suitable message.**

**Algorithm:**

1. Insert function:
   1. temp->data = n;
   2. temp->next = head;
   3. head = temp;
2. Search function:
   1. index = 0;
   2. for (ptr = head; ptr != NULL; ptr = ptr->next)
      1. index++;
      2. if (ptr->data == element)
         1. print "Element found at position: index”
         2. return;
   3. print "Data not found"

**Test Case Table:**

**For a given input list as attached below**

|  |  |  |
| --- | --- | --- |
| Search element | Position | Pass/Fail |
| 43 | 7 | Pass |
| 86 | 2 | Pass |
| 2 | Data not found | Pass |

**Program:**

#include <stdio.h>

#include <stdlib.h>

#include <time.h>

#include <ctype.h>

struct node{

    int data;

    struct node \*next;

}\*head=NULL, \*ptr=NULL;

void insertbeg(int n){

    struct node \*temp=NULL;

    temp = (struct node \*)malloc(sizeof(struct node));

    if(temp == NULL){

        printf("Memory can't be allocated\n");

        return;

    }

    temp->data = n;

    temp->next = head;

    head = temp;

}

void search(int s){

    int index = 0;

    for (ptr = head; ptr != NULL; ptr = ptr->next)

    {

        index++;

        if (ptr->data == s)

        {

            printf("Element found at position: %d\n", index);

            return;

        }

    }

    printf("Data not found\n");

}

void display(){

    for (ptr = head; ptr != NULL; ptr = ptr->next)

    {

        printf("%d ", ptr->data);

    }

}

int main(){

    srand(time(0));

    printf("Number of elements to insert: ");

    int n;

    scanf("%d", &n);

    for (int i = 0; i < n; i++)

    {

        insertbeg(rand() % 100);

    }

    printf("Linked list: ");

    display();

    char choice;

    printf("\nSample data inserted..\n\n");

    while(1){

        printf("Do you want to search data (y/n): ");

        scanf("%c", &choice);

        switch (tolower(choice))

        {

        case 'y':

            printf("Enter number you want to search (0-100): ");

            int data;

            scanf("%d", &data);

            search(data);

            break;

        case 'n':

            goto Exit\_1;

            break;

        default:

            printf("Invalid selection.\n");

            break;

        }

        while((getchar()) != '\n');

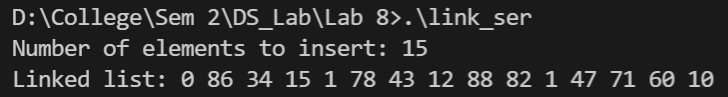
    }

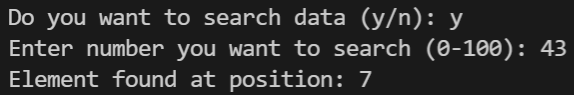
    Exit\_1:printf("\n");

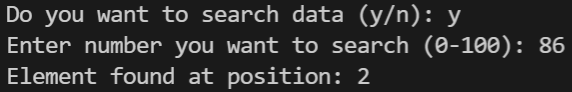
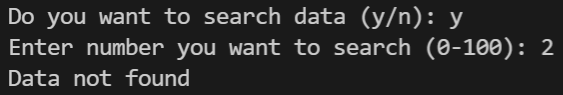
    return 0;

}

**Screenshot of compilation and execution:**

****

****

**** ****

1. **Activity: Store n numbers in a Hash Table implemented with array of Linked Lists. Perform search operation for a given number and report success or failure in the form of suitable message.**

**Algorithm:**

1. Hash function:
   1. Get data
   2. H = (int)data/10
2. Insert function:
   1. index = hash(key)
   2. Temp->data = key, temp->next = NULL
   3. If table[index] == NULL, table[index] = temp
   4. Else
      1. For(ptr = table[index]; ptr->next != NULL; ptr = ptr->next)
      2. Ptr->next = temp
3. Search function:
   1. index = hash(key), comp++
   2. if table[index] == NULL, print data not found
   3. count = 1
   4. For (ptr = table[index]; ptr->data != key; ptr = ptr->next, count++)
      1. Comp ++
      2. If(ptr->next == NULL)
         1. Comp++
         2. print data not found
   5. Print element found at index (count) of list in index (index)

**Test Case Table:**

**For a given input list as attached below**

|  |  |  |
| --- | --- | --- |
| Search element | Index | Pass/Fail |
| 67 | (6, 3) | Pass |
| 76 | (7, 1) | Pass |
| 55 | (5, 2) | Pass |
| 72 | Data not Found | Pass |

**Program:**

#include <stdio.h>

#include <stdlib.h>

#include <time.h>

#include <ctype.h>

struct Node{

    int data;

    struct Node \*next;

} \*ptr = NULL, \*table[10] = {NULL};

//Hash function

int hash(int key){

    int h;

    h = key / 10;

    return h;

}

void insert(int key){

    int index = hash(key);

    struct Node \*temp = (struct Node \*)malloc(sizeof(struct Node));

    temp->data  = key;

    temp->next = NULL;

    if(table[index] == NULL){

        table[index] = temp;

    }

    else{

        for(ptr = table[index]; ptr->next != NULL; ptr = ptr->next);

        ptr->next = temp;

    }

}

void search(int key){

    int index = hash(key), comp = 0;

    if (table[index] == NULL)

    {

        printf("Data Not Found\n");

        return;

    }

    int count = 1;

    for (ptr = table[index]; ptr->data != key; ptr = ptr->next, count ++){

        comp++;

        if (ptr->next == NULL)

        {

            comp++;

            printf("Comparision made: %d\n", comp);

            printf("Data Not Found\n");

            return;

        }

    }

    comp++;

    printf("Comparision made: %d\n", comp);

    printf("Element found at position %d of list in index %d\n", count, index);

}

void print\_table(){

    printf("Index\tData\n");

    for (int i = 0; i < 10; i++)

    {

        printf("%d\t", i);

        for (ptr = table[i]; ptr != NULL; ptr = ptr->next)

        {

            printf("%d\t", ptr->data);

        }

        printf("\n");

    }

}

int main(){

    srand(time(0));

    printf("Number of elements to insert: ");

    int n;

    scanf("%d", &n);

    for (int i = 0; i < n; i++)

    {

        insert(rand() % 100);

    }

    print\_table();

    char choice;

    printf("Sample data inserted..\n\n");

    while(1){

        while((getchar()) != '\n');

        printf("Do you want to search data (y/n): ");

        scanf("%c", &choice);

        switch (tolower(choice))

        {

        case 'y':

            printf("Enter number you want to search (0-100): ");

            int data;

            scanf("%d", &data);

            search(data);

            break;

        case 'n':

            goto Exit\_1;

            break;

        default:

            printf("Invalid selection.\n");

            break;

        }

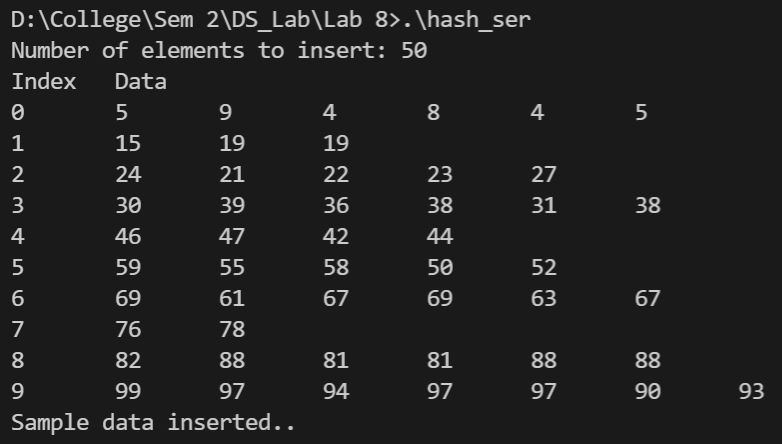
    }

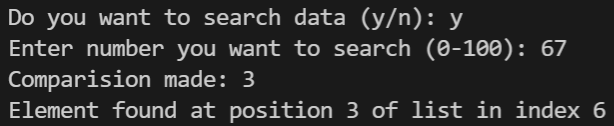
    Exit\_1:printf("\n");

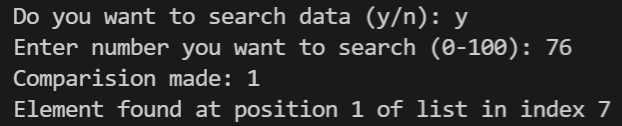
    return 0;

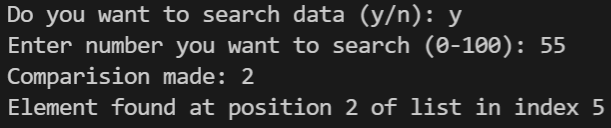
}

**Screenshot of compilation and execution:**

****

****

****

****