NAME : P . VINEETHA

DAY-1

DATE : 24-07-2024

1. Write a program of linear search .

```
  #include <stdio.h>
void LinearSearch(int arr[], int len, int item)
{
	for(int i=0;i < len;i++)
	{
		if(arr[i] == item)
		{
		printf("%d Found at index %d", item, i);
		return;
		}
	}
	printf("Not Found");
}
int main()
{
	int arr[] = {10, 20, 30, 40, 50};
      // calculating length of array
	int len = sizeof(arr)/sizeof(arr[0]);
// item to be searched
	int item = 40;
```

```
    LinearSearch(arr, len, item);
return 0;
}
```

Output : 40 Found at index 3

2. Write a program for binary search .

```
#include <stdio.h>
int binarySearch(int arr[], int left, int right, int item){
     if (right >= left){
          int mid = left + (right - left)/2;
          if (arr[mid] == item)
               return mid;
               if (arr[mid] > item)
               return binarySearch(arr, left, mid-1, item);
          else
               return binarySearch(arr, mid+1, right, item);
     }
     else
          return -1;
}
int main(){
     int arr[8] = {10, 20, 30, 40, 50, 60, 70, 80};
     int n = sizeof(arr) / sizeof(arr[0]);
     int item = 70;
```

```
    int position = binarySearch(arr, 0, n-1, item);
if(position == -1)
        printf("%d Not Found",item);
    else
        printf("%d Found at index : %d",item, position);
return 0;
}
```

Output : 30 Found at index : 2

3. write a c program to implement the following operations Traverse,Search,Insert,Delete,Update .

```
 #include <stdio.h>
#include <stdlib.h>
// Node structure
typedef struct Node {
    int data;
    struct Node* next;
} Node;
Node* createNode(int data) {
    Node* newNode = (Node*)malloc(sizeof(Node));
    if (!newNode) {
        printf("Memory error\n");
        return NULL;
    }
```

```
    newNode->data = data;
    newNode->next = NULL;
    return newNode;
}
void traverse(Node* head) {
    Node* current = head;
    while (current != NULL) {
        printf("%d -> ", current->data);
        current = current->next;
    }
    printf("NULL\n");
}
Node* search(Node* head, int key) {
    Node* current = head;
    while (current != NULL) {
        if (current->data == key) {
            return current;
        }
        current = current->next;
    }
    return NULL;
}
void insert(Node** head, int data) {
    Node* newNode = createNode(data);
    newNode->next = *head;
```

```
        *head = newNode;
}
void deleteNode(Node** head, int key) {
        Node* temp = *head;
        Node* prev = NULL;
if (temp != NULL && temp->data == key) {
                *head = temp->next;
                free(temp);
                return;
        }
    while (temp != NULL && temp->data != key) {
                prev = temp;
                temp = temp->next;
        }
if (temp == NULL) return;
  prev->next = temp->next;
        free(temp);
}
void update(Node* head, int oldData, int newData) {
        Node* nodeToUpdate = search(head, oldData);
        if (nodeToUpdate) {
                nodeToUpdate->data = newData;
        } else {
                printf("Node with data %d not found.\n", oldData);
        }
```

```
}
int main() {
        Node* head = NULL;
insert(&head, 10);
        insert(&head, 20);
        insert(&head, 30);
printf("Linked List: ");
        traverse(head);
  int key = 20;
        Node* foundNode = search(head, key);
        if (foundNode) {
                printf("Node with data %d found.\n", foundNode->data);
        } else {
                printf("Node with data %d not found.\n", key);
        }
printf("Updating node with data 20 to 25.\n");
        update(head, 20, 25);
        printf("Linked List after update: ");
        traverse(head);
  printf("Deleting node with data 10.\n");
        deleteNode(&head, 10);
        printf("Linked List after deletion: ");
        traverse(head);
  return 0;
}
```

Output :

Linked List: 30 -> 20 -> 10 -> NULL

Updating node with data 20 to 25.

Linked List after update: 30 -> 25 -> 10 -> NULL

Deleting node with data 10.

Linked List after deletion: 30 -> 25 -> NULL

4. Write a recursive function to calculate the factorial of the number .

```
#include <stdio.h>
int factorial(int n) {
	if (n == 0) {
		return 1;
	} else {
		return n * factorial(n - 1);
	}
}
int main() {
	int number = 5;
	int result = factorial(number);
	printf("Factorial of %d = %d", number, result);
	return 0;
}
```

Output : Factorial of 5 = 120

5. Write a program to find the duplicate elements in an array .

```
#include <stdio.h>

void findDuplicates(int arr[], int size) {
	for (int i = 0; i < size; i++) {
		for (int j = i + 1; j < size; j++) {
			if (arr[i] == arr[j]) {
				printf("Duplicate Element: %d\n", arr[j]);
			}
		}
	}
}

int main() {
	int arr[] = {1, 2, 3, 4, 2, 7, 8, 8, 3};
	int size = sizeof(arr) / sizeof(arr[0]);
printf("Duplicate elements in the array are: \n");
	findDuplicates(arr, size);
return 0;
}
```

Output :

```
Duplicate elements in the array are:
Duplicate Element: 2
Duplicate Element: 3
Duplicate Element: 8
```

6. write a program to find max and min from an array elements.

```
#include <stdio.h>

int main() {

	int arr[] = {10, 5, 8, 20, 15};

	int n = sizeof(arr) / sizeof(arr[0]);

	int max = arr[0];

	int min = arr[0];

for (int i = 1; i < n; i++) {

		if (arr[i] > max) {

			max = arr[i];

		}

		if (arr[i] < min) {

			min = arr[i];

		}

	}

printf("Maximum element in the array: %d\n", max);

	printf("Minimum element in the array: %d\n", min);

return 0;

}
```

Output : Maximum element in the array: 20

Minimum element in the array: 5

7 . Given a number n. the task is to print the Fibonacci series and the sum of the series using recursion .

```
#include <stdio.h>

int fibonacci(int n) {
    if (n <= 1)
        return n;
    return fibonacci(n - 1) + fibonacci(n - 2);
}

int main() {
    int n = 10;
    int sum = 0;
printf("Fibonacci series:\n");
    for (int i = 0; i < n; i++) {
        printf("%d, ", fibonacci(i));
        sum += fibonacci(i);
    }
printf("\nSum: %d\n", sum);
return 0;
}
```

Output : Fibonacci series:

0, 1, 1, 2, 3, 5, 8, 13, 21, 34,

Sum: 88

8 . You are given an array arr in increasing order. Find the element x from arr using binary search.

```
#include <stdio.h>
```

```
int binarySearch(int arr[], int left, int right, int x) {
        while (left <= right) {
                int mid = left + (right - left) / 2;
if (arr[mid] == x)
                        return mid;
                if (arr[mid] < x)
                        left = mid + 1;
                else
                        right = mid - 1;
        }
    return -1;
}
int main() {
        int arr[] = {1, 5, 6, 7, 9, 10};
        int n = sizeof(arr) / sizeof(arr[0]);
        int x = 6;
        int result = binarySearch(arr, 0, n - 1, x);
if (result == -1)
                printf("Element not found\n");
        else
                printf("Element found at location %d\n", result);
return 0;
}
```

Output:

```
Element found at location 2
```