

## LIST OF PROGRAMS

### 1.Linear search

```
#include <stdio.h>

int main()
{
    int array[100], search, c, n;

    printf("Enter number of elements in array\n");
    scanf("%d", &n);

    printf("Enter %d integer(s)\n", n);

    for (c = 0; c < n; c++)
        scanf("%d", &array[c]);

    printf("Enter a number to search\n");
    scanf("%d", &search);
    for (c = 0; c < n; c++)
    {
        if (array[c] == search)    /* If required element is found */
        {
            printf("%d is present at location %d.\n", search, c+1);
            break;
        }
    }
    if (c == n)
        printf("%d isn't present in the array.\n", search);
}
```

```
    return 0;
}
```

```
Enter number of elements in array
5
Enter 5 integer(s)
1
2
3
4
5
Enter a number to search
2
2 is present at location 2.

=== Code Execution Successful ===
```

## 2.BINARY SEARCH

```
#include <stdio.h>

int main()
{
    int c, first, last, middle, n, search, array[100];

    printf("Enter number of elements\n");
    scanf("%d", &n);

    printf("Enter %d integers\n", n);

    for (c = 0; c < n; c++)
        scanf("%d", &array[c]);
```

```
printf("Enter value to find\n");  
scanf("%d", &search);  
  
first = 0;  
last = n - 1;  
middle = (first+last)/2;  
  
while (first <= last) {  
    if (array[middle] < search)  
        first = middle + 1;  
    else if (array[middle] == search) {  
        printf("%d found at location %d.\n", search, middle+1);  
        break;  
    }  
    else  
        last = middle - 1;  
  
    middle = (first + last)/2;  
}  
if (first > last)  
    printf("Not found! %d isn't present in the list.\n", search);  
  
return 0;  
}
```

```
Enter number of elements
5
Enter 5 integers
2
3
4
5
6
Enter value to find
4
4 found at location 3.

=== Code Execution Successful ===
```

### 3.TRAVERSE

```
#include<stdio.h>

void main()
{
    int a[]={2,4,6,8,10};
    int n,i;
    n=sizeof(a)/sizeof(a[0]);
    for(i=0;i<n;i++)
    {
        printf("%d",a[i]);
        printf(" ");
    }
}
```

```
/tmp/yAVU1RKCI4.o
2 4 6 8 10

=== Code Exited With Errors ===
```

#### 4.SEARCH

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

```
void main()
```

```
{
```

```
    int a[]={1,2,3,4,5,6};
```

```
    int n,i,s,c=0;
```

```
    n=sizeof(a)/sizeof(a[0]);
```

```
    printf("enter element:");
```

```
    scanf("%d",&s);
```

```
    for(i=0;i<n;i++)
```

```
    {
```

```
        if(a[i]==s)
```

```
        {
```

```
            c=1;
```

```
            break;
```

```
        }
```

```
    }
```

```
    if(c==1)
```

```
    {
```

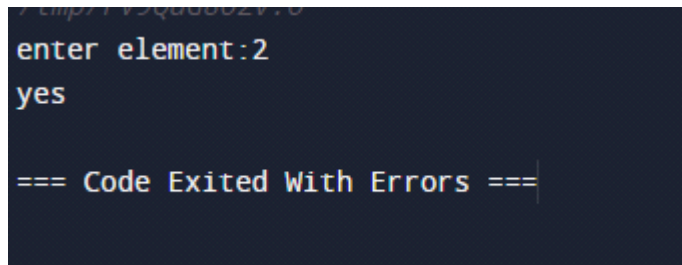
```
        printf("yes");
```

```
    }
```

```
    else
```

```
    {
```

```
        printf("no");  
    }  
}
```



A terminal window with a dark background. It shows the prompt 'emp/153488821:0', the user input 'enter element:2', and the program output 'yes'. Below this, a red error message is displayed: '=== Code Exited With Errors ==='.

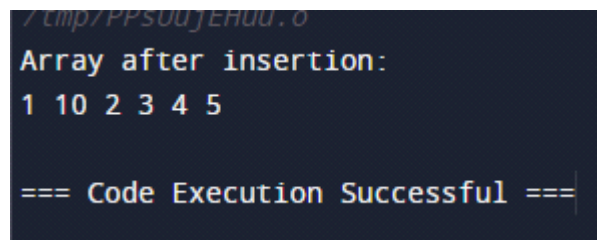
```
emp/153488821:0  
enter element:2  
yes  
  
=== Code Exited With Errors ===
```

## 5.INSERT

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

```
int main() {  
    int array[5] = {1, 2, 3, 4, 5};  
    int position = 2;  
    int newValue = 10;  
    int n = 5;  
  
    for (int i = n - 1; i >= position - 1; i--) {  
        array[i + 1] = array[i];  
    }  
  
    array[position - 1] = newValue;  
    n++;  
  
    printf("Array after insertion:\n");  
    for (int i = 0; i < n; i++) {  
        printf("%d ", array[i]);  
    }  
}
```

```
    return 0;
}
```



```
7/emp7PPS00JEH00.0
Array after insertion:
1 10 2 3 4 5

=== Code Execution Successful ===
```

## 6. Writing a recursive function to calculate the factorial of a 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

/tmp/aqGz9EtaBs.o

Factorial of 5 = 120

=== Code Execution Successful ===

## 7. Write a C Program to find duplicate element in an array

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

```
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: ");  
    for (int i = 0; i < size; i++) {  
        for (int j = i + 1; j < size; j++) {  
            if (arr[i] == arr[j]) {  
                printf("%d ", arr[j]);  
                break;  
            }  
        }  
    }  
  
    return 0;  
}
```



## Output

```
/tmp/Upws44HJqR.o  
Duplicate elements in the array are: 2 3 8  
=== Code Execution Successful ===
```

### 8. Write a C Program to find Max and Min from an array elements

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

```
int main() {  
    int arr[] = {10, 5, 8, 20, 2};  
    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
/tmp/gtQoAntEf0.o
Maximum element in the array: 20
Minimum element in the array: 2

=== Code Execution Successful ===
```

**9. 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, sum = 0;

    printf("Enter the number of terms: ");
    scanf("%d", &n);

    printf("Fibonacci Series: ");
    for (int i = 0; i < n; i++) {
        printf("%d ", fibonacci(i));
        sum += fibonacci(i);
    }
```

```
printf("\nSum of Fibonacci Series: %d", sum);
```

```
return 0;
```

```
}
```

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

```
#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[] = {2, 4, 6, 8, 10, 12, 14, 16};
```

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

```
    int x = 10;
```

```
int result = binarySearch(arr, 0, n - 1, x);

if (result == -1)
    printf("Element not found\n");
else
    printf("Element found at index %d\n", result);

return 0;
}
```

### Output

```
/tmp/uYIj5C47HX.o
Element found at index 4

=== Code Execution Successful ===
```