

1) Write a program to find the largest & smallest element in an assignment

→ #include <stdio.h>

int main () {

int n, i;

int arr [100];

printf ("Enter number of elements");

scanf ("%d", &n);

printf ("Enter elements : ");

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

scanf ("%d", &arr[i]);

}

int max = arr[0];

int min = arr[0];

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

if (arr[i] > max)

max = arr[i];

if (arr[i] < min)

min = arr[i];

}

printf ("largest element = %d\n", max);

printf ("smallest element = %d\n", min);

return 0;

}

output

Enter number of elements 3

Enter elements: 1 2 3

largest element = 3

smallest element = 2

```
1 #include <stdio.h>
2 int main() {
3     int n, i;
4     int arr [100];
5     printf("Enter number of elements: ");
6     scanf("%d", &n);
7     printf("Enter elements: \n");
8     for (i = 0; i < n; i++) {
9         scanf ("%d",&arr[i]);
0     }
1
2     int max = arr[0];
3     int min = arr [0];
4
5     for (i = 1; i < n; i++) {
6         if (arr [i] > max)
7             max = arr[i];
8
9         if (arr[i] < min)
0             min = arr[i];
1     }
2     printf("Largest element = %d\n", max);
3     printf("Smallest element = %d\n",min);
4     return 0;
5 }
```

Enter number of elements: 3
Enter elements:
1 2 3
Largest element = 3
Smallest element = 1

==== Code Execution Successful ===

3. Write a program to find the factorial of a number using recursion

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

```
int factorial (int n) {  
    if (n == 0 || n == 1)  
        return 1;
```

```
else
```

```
    return n * factorial (n - 1);
```

```
y
```

```
int main () {
```

```
    int num;
```

```
    printf ("Enter a number : ");
```

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

```
    printf ("Factorial = %d", factorial (num));
```

```
    return 0;
```

```
3
```

output

Enter a number : 10

Factorial = 3628800

```
1 #include <stdio.h>
2
3 int factorial (int n) {
4     if (n == 0 || n == 1)
5         return 1;
6     else
7         return n * factorial(n - 1);
8 }
9
10 int main() {
11     int num;
12
13     printf("Enter a number: ");
14     scanf("%d", &num);
15
16     printf("Factorial = %d", factorial(num));
17
18     return 0;
19 }
```

```
Enter a number: 10
Factorial = 3628800
==== Code Execution Successful ===
```

3 Write a program to find the n^{th} Fibonacci number using recursion

```
# include <stdio.h>
int fibonaci (int n) {
    if (n == 0)
        return 0;
    else if (n == 1)
        return 1;
    else
        return fibonaci(n-1) + fibonaci(n-2);
}
int main () {
    int n;
    printf ("Enter n: ");
    scanf ("%d", &n);
    printf ("Fibonacci number = %d",
           fibonaci (n));
    return 0;
}
```

Output

Enter Enter n = 10

Fibonacci number = 55

```
#include <stdio.h>
int fibonacci(int n) {
    if(n == 0)
        return 0;
    else if(n == 1)
        return 1;
    else
        return fibonacci(n-1) + fibonacci(n-2);
}
int main() {
    int n;
    printf("Enter n: ");
    scanf ("%d",&n);
    printf("Fibonacci number = %d", fibonacci(n));
    return 0;
}
```

```
Enter n: 10
Fibonacci number = 55
== Code Execution Successful ==
```

4

```

#include <stdio.h>
#include <stdlib.h>
struct Node {
    int data;
    struct Node *next;
};

int main() {
    struct Node *head, *first, *second;
    head = (struct Node *) malloc(sizeof(struct Node));
    first = (struct Node *) malloc(sizeof(struct Node));
    second = (struct Node *) malloc(sizeof(struct Node));
    head->data = 100;
    head->next = first;
    first->data = 200;
    first->next = second;
    second->data = 300;
    second->next = NULL;
    struct Node *temp = head;
    printf("Linked list = ");
    while (temp != NULL) {
        printf("%d->%d", temp->data);
        temp = temp->next;
    }
    printf("NULL");
    return 0;
}

```

4

```
#include <stdio.h>
#include <stdlib.h>
struct Node {
    int data;
    struct Node*next;
};
int main() {
    struct Node *head, *first, *second;
    head=(struct Node*)malloc(sizeof(struct Node));
    first=(struct Node*)malloc(sizeof(struct Node));
    second=(struct Node*)malloc(sizeof(struct Node));
    head->data=100;
    head->next=first;
    first->data=200;
    first->next=second;
    second->data=300;
    second->next=NULL;
    struct Node *temp=head;
    printf("Linked List:");
    while(temp!=NULL){
        printf("*d-", temp->data);
        temp = temp->next;
    }
    printf( "NULL");
    return 0;
}
```

Linked List:*d-,*d-*d->NULL

== Code Execution Successful ==