Page No:
Date.

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 :\n");
    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
```

```c
#include <stdio.h>
int main() {
    int n, i;
    int arr [100];
    printf("Enter number of elements: ");
    scanf("%d", &n);
    printf("Enter elements: \n");
    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;
}
```

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


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

Page No:
Date.

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);
}
int main() {
    int num;
    Print f("Enter a number: ");
    scan f ("%d", &num);
Print ("Factorial = %d", factorial (num));
    return 0;
}
```

output

```
Enter a number : 10
Factorial = 3628800
```

```c
#include <stdio.h>

int factorial (int n) {
if (n == 0 || n == 1)
    return 1;
else
    return n * factorial(n - 1);
}

int main() {
    int num;

    printf("Enter a number: ");
    scanf("%d", &num);

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

    return 0;
}
```

```
Enter a number: 10
Factorial = 3628800

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

Page No:
Date.

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

```
#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;
}
```

Output

Enter n = 10
Fibonacci number = 55

```c
#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 ===
```

Page No:
Date.

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 (size of (struct
        first = (struct Node *) malloc (size of (struct Node)
        second = (struct node *) malloc (size of (struct Node));
head -> data = 100;
head -> next = first;
first -> data = 200;
first -> next = second;
second -> data = 300;
second -> next = null;
 struct Node * temp = head;
    Print f ("Linked list =");
  while (temp != Null) {
     print f ("%d -> " temp -> data);
        temp = temp -> next;
    } {
        Print f ("NULL");
        return 0;
    }
```

```c
#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 ===
```