

ASSIGNMENT-9

a) Functions without arguments and without return type.

☐ Check whether the year is Leap year.

PROGRAM:

```
#include<stdio.h> void leapYear(){ int year;
printf("Enter a year :: ");
scanf("%d",&year);

if(year % 4 == 0 && year % 100 != 0 || year % 400 == 0){ printf("%d is a Leap
year.",year);
}
else{
printf("%d is not a Leap year.",year);

}
}

int main(){
leapYear(); return 0;
}
```

OUTPUT:

Enter a year :: 2004 2004 is a Leap year.

Count number of digits in a number.

PROGRAM:

```
#include<stdio.h> void countNum(){ int num,d,c=0,a;
printf("Input a number :: "); scanf("%d",&num);
a=num; while(num>0){ d = num % 10;
c++;
```

```

num /= 10;
}
printf("%d number contain %d digits",a,c);
}
int main(){
countNum(); return 0;
}

```

OUTPUT:

Input a number :: 456987 456987 number contain 6 digits

b) Functions without arguments and with return type.

☐ check Armstrong number or not

PROGRAM:

```

#include<stdio.h>

int checkArmstrong(){ int num,d,c=0,a,s=0;
printf("Input a number :: "); scanf("%d",&num);
a=num; while(num>0){ d = num % 10;
s = s + d * d * d; num /= 10;
}
if(s == a){ return 1;
}
else{ return 0;
}
}

int main(){
int check;

```

```
check = checkArmstrong(); if(check == 1){  
printf("This is a Armstrong number");  
}  
else{  
printf("This is not a Armstrong number");  
}  
return 0;  
}
```

OUTPUT:

Input a number :: 153

This is a Armstrong number

□ Convert temperature Fahrenheit to Celsius.

PROGRAM:

```
#include<stdio.h>  
  
float TemperatureConv(){ float temp,result;  
printf("Input a temprature in Fahrenheit:: "); scanf("%f",&temp);  
result = (temp - 32) * 0.5; return result;  
}  
  
int main(){ float celsius;  
celsius = TemperatureConv(); printf("%g c",celsius);  
return 0;  
}
```

OUTPUT:

Input a temprature in Fahrenheit:: 67

17.5 c

c) Functions with arguments and without return type.

☐ Check prime number or not.

PROGRAM:

```
#include<stdio.h>

void checkPrime(int n){ int d=1,c=0;
while(d <= n){ if(n % d == 0){ c++;
} d++;
}
if(c == 2){
printf("%d is a prime number.",n);
}
else{
printf("%d is not a prime number.",n);
}
}

int main(){ int num;
printf("Enter a number :: "); scanf("%d",&num);
checkPrime(num);
return 0;
}
```

OUTPUT:

Enter a number :: 5 5 is a prime number.

☐ Find all roots of the quadratic equation.

PROGRAM:

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

```

void roots(int a,int b,int c){ int d;
d = b * b - 4 * a * c;
if(d > 0){
printf("roots are real and different");
}
else if(d == 0){
printf("roots are real and equal");
}
else{
printf("roots are complex and different");
}
}

int main(){ int a,b,c;
printf("Enter the value of 'a' 'b' 'c' :: ");
scanf("%d%d%d",&a,&b,&c); roots(a,b,c);
return 0;
}

```

OUTPUT:

```

Enter the value of 'a' 'b' 'c' :: 2 2 2
roots are complex and different

```

d) Functions with arguments and with return type.

☐ Check perfect or abundant or deficient number.

PROGRAM:

```

#include<stdio.h>

int checkNum(int num){ int sum=0,d=1;

```

```

while(d <= num){ if(num % d == 0){ sum += d;
} d++;
}
if((sum - num) == num){ return 1; //perfect no
}
else if((sum - num) > num){ return 2; //abundant
}
else if(sum < (2 * num)){ return 3;
}
}
int main(){ int res,n;
printf("Enter a number :: "); scanf("%d",&n);
res = checkNum(n); if(res == 1){
printf("%d is a perfect number.",n);
}
else if(res == 2){
printf("%d is a abundant number.",n);
}
else if(res == 3){
printf("%d is a deficient number.",n);
}
return 0;
}

```

OUTPUT:

Enter a number :: 12

12 is a abundant number.

Enter a number :: 6

6 is a perfect number.

Enter a number :: 21

21 is a deficient number.

☐ Calculate factorial of a number.

PROGRAM:

```
#include<stdio.h> int fact(int num){ int mul = 1; while(num >= 1){ mul *=  
num;  
num--;  
}  
return mul;  
}  
  
int main(){ int n,fa;  
printf("Enter a number :: "); scanf("%d",&n);  
fa = fact(n);  
printf("%d factorial is %d",n,fa); return 0;  
}
```

OUTPUT:

Enter a number :: 5 5 factorial is 120

e) Function return Multiple values

☐ Largest and Smallest of five numbers.

PROGRAM:

```
#include<stdio.h>  
  
void compareNum(int *greater,int *smaller){ int arr[5],i,j,temp=0;
```

```

for(i=0;i<5;i++){
printf("Enter a number :: "); scanf("%d",&arr[i]);
}
for(i=0;i<5;i++){ for(j=0;j<5;j++){
if(arr[i] < arr[j]){ temp = arr[i]; arr[i] = arr[j]; arr[j] = temp;
}
}
}
*greater = arr[4];
*smaller = arr[0];
}
void main(){ int g,s;
compareNum(&g,&s);
printf("Largest number = %d \nSmallest number = %d",g,s);
}

```

OUTPUT:

Enter a number :: 5

Enter a number :: 6

Enter a number :: 3

Enter a number :: 2

Enter a number :: 4

Largest number = 6

Smallest number = 2

☐ Find Simple interest and compound interest.

PROGRAM:

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



```

void interestCal(float* si,float* ci){ float p,r,t;
printf("Enter principle :: "); scanf("%f",&p);
printf("Enter rate :: ");
scanf("%f",&r);
printf("Enter time :: ");
scanf("%f",&t);
*si =(p * r * t) / 100;
*ci = p * (pow((1 + r / 100),t) - 1);
}
void main(){ float s,c;
interestCal(&s,&c);
printf("Simple interest = %.3f \nCompound interest = %.3f",s,c);
}

```

OUTPUT:

```

Enter principle :: 4000
Enter rate :: 11
Enter time :: 3
Simple interest = 1320.000
Compound interest = 1470.524

```

f) Nesting of Functions.

- Print the sum of series $1 + \frac{1}{2} + \frac{1}{3} + \frac{1}{4} + \dots + \frac{1}{N}$.

PROGRAM:

```

#include<stdio.h> void input(){
int n;
printf("Enter a number :: "); scanf("%d",&n);

```

```

void series(int num){ float res=1;
int i=1; for(;i<=num;i++){
printf("1 + %d / ",i); res /= 1 + i;
}
printf("= %g",res);
}
series(n);
}
int main(){ input();
return 0;
}

```

OUTPUT:

Enter a number :: 4

1 + 1 / 1 + 2 / 1 + 3 / 1 + 4 / = 0.00833333

□ Reverse a number.

PROGRAM:

```

#include<stdio.h> void input(){
int n;
printf("Enter a number :: "); scanf("%d",&n);
void reverse(int num){ int d,rev=0;
printf("Reverse order :: "); while(num>0){
d = num % 10; rev = rev * 10 + d; num /=10;
}
printf("%d",rev);
}
reverse(n);

```

```
}  
int main(){ input();  
return 0;  
}
```

OUTPUT:

Enter a number :: 4896325

Reverse order :: 5236984

g) Recursive Functions.

□ To Print Fibonacci Series.

PROGRAM:

```
#include<stdio.h> int sum =0,i=0,j=1; int input(int n){  
if(n>0){  
sum = i + j; printf("%d ",sum);  
i = j;  
j = sum;  
return input(n-1);  
}  
}  
  
int main(){ int num;  
printf("Enter a number :: "); scanf("%d",&num);  
printf("Fibonacci series upto %d terms = 0 1 ",num); input(num);  
return 0;  
}
```

OUTPUT:

Enter a number :: 10

Fibonacci series upto 10 terms = 0 1 1 2 3 5 8 13 21 34 55 89

□ To print even or odd numbers in given range.

PROGRAM:

```
#include<stdio.h> int input(int n){
if(n>0){
if(n % 2 == 0){
printf("even number= %d \n",n);
}
else{
printf("odd number= %d \n",n);
}
return input(n-1);
}
}

int main(){ int num;
printf("Enter a number :: ");
scanf("%d",&num); input(num);
return 0;
}
```

Output:

Enter a number :: 6

even number= 6

odd number= 5

even number= 4

odd number= 3

even number= 2

odd number= 1

h) Passing 1D Array in Functions.

□ Reverse the elements of an array.

PROGRAM:

```
#include<stdio.h>

void reverse(int n[]){ int i=4;
printf("Array in reverse order :: "); for(;i>=0;i--){
printf("%d ",n[i]);
}
}

int main(){ int arr[5],i=0;
for(;i<5;i++){
printf("Enter a number :: "); scanf("%d",&arr[i]);
}
reverse(arr);
return 0;
}
```

OUTPUT:

Enter a number :: 6

Enter a number :: 4

Enter a number :: 5

Enter a number :: 9

Enter a number :: 8

Array in reverse order :: 8 9 5 4 6

- Find the fourth largest and Third smallest element in an array.

PROGRAM:

```
#include<stdio.h> void number(int n[]){
int i=0,j,temp=0;
for(i=0;i<10;i++){ for(j=0;j<10;j++){
if(n[i] < n[j]){ temp = n[j]; n[j] = n[i]; n[i] = temp;
}
}
}
printf("In this given array\nFourth largest = %d\nThird smallest
= %d ",n[6],n[2]);
}
int main(){
int arr[10],i=0;
for(;i<10;i++){
printf("Enter a number :: "); scanf("%d",&arr[i]);
}
number(arr);
return 0;
}
```

OUTPUT:

Enter a number :: 3

Enter a number :: 2

Enter a number :: 1

Enter a number :: 6

Enter a number :: 5

Enter a number :: 4

Enter a number :: 9

Enter a number :: 8

Enter a number :: 7

Enter a number :: 10

In this given array Fourth largest = 7 Third smallest = 3

i) Passing 2D Array in Functions.

□ Sum of upper triangular and lower triangular elements of mxm array.

PROGRAM:

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

```
void triangleSum(int n[3][3]){ int i,j,sum=0;
```

```
for(i=0;i<3;i++){ for(j=0;j<3;j++){
```

```
printf("%d ",n[i][j]);
```

```
}
```

```
printf("\n");
```

```
}
```

```
for(i=0;i<3;i++){ for(j=i;j<3;j++){ sum += n[i][j];
```

```
}
```

```
}
```

```
printf("sum of upper triangle = %d\n",sum); sum = 0;
```

```
for(i=0;i<3;i++){
```

```
for(j=i;j<3;j++){ sum += n[j][i];
```

```
}
```

```
}
```

```

printf("sum of lower triangle = %d\n",sum);
}
int main(){
int arr[3][3],i,j;
for(i=0;i<3;i++){ for(j=0;j<3;j++){
printf("Enter a number :: "); scanf("%d",&arr[i][j]);
}
}
triangleSum(arr);
return 0;
}

```

Output:

Enter a number :: 1

Enter a number :: 2

Enter a number :: 3

Enter a number :: 4

Enter a number :: 5

Enter a number :: 6

Enter a number :: 8

Enter a number :: 9

1 2 3

4 5 6

7 8 9

sum of upper triangle = 26

sum of lower triangle = 34

□ Perform matrix multiplication between two mxn array.

PROGRAM:

```
#include<stdio.h>

void matrixMul(int n[][2],int m[][2]){ int mul[2][2],i,j,k;
printf("first matrix ::\n"); for(i=0;i<2;i++){
for(j=0;j<2;j++){ printf("%d ",n[i][j]);
}
printf("\n");
}
printf("second matrix ::\n"); for(i=0;i<2;i++){
for(j=0;j<2;j++){ printf("%d ",m[i][j]);
}
printf("\n");
}
printf("Matrix multiplication :: \n");
for(i=0;i<2;i++){ for(j=0;j<2;j++){
mul[i][j] = 0; for(k=0;k<2;k++){
mul[i][j] += n[i][k] * m[k][j];
}
}
}
for(i=0;i<2;i++){ for(j=0;j<2;j++){
printf("%d ",mul[i][j]);
}
printf("\n");
}
}
```

```

int main(){
int first[2][2],second[2][2],i,j; printf("Enter element in first matrix ::\n");
for(i=0;i<2;i++){
for(j=0;j<2;j++){
printf("Enter a number :: ");
scanf("%d",&first[i][j]);
}
}
printf("Enter element in second matrix ::\n"); for(i=0;i<2;i++){
for(j=0;j<2;j++){
printf("Enter a number :: "); scanf("%d",&second[i][j]);
}
}
matrixMul(first,second);
return 0;
}

```

Output:

Enter element in first matrix ::

Enter a number :: 1

Enter a number :: 2

Enter a number :: 3

Enter a number :: 4

Enter element in second matrix ::

Enter a number :: 4

Enter a number :: 3

Enter a number :: 2

Enter a number :: 1

first matrix ::

1 2

3 4

second matrix ::

4 3

2 1

Matrix multiplication :: 8 5

20 13

j) Passing Strings in Functions.

□ To read a string and prints if it is a palindrome or not.

PROGRAM:

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

```
void stringPalindrome(char v[]){ int c=0,i=0,j;
```

```
while(v[i] != '\0'){ c++;
```

```
i++;
```

```
} i=0;
```

```
j=c-1; c=0;
```

```
while(v[i] != '\0'){
```

```
if(v[i] != v[j]){ c++;
```

```
break;
```

```
} i++; j--;
```

```
}
```

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

```
printf("%s is not a palindrome string",v);
```

```
}  
else{  
printf("%s is a palindrome string",v);  
}  
}  
  
int main(){ char s[15];  
printf("Enter a string :: "); scanf("%s",s);  
stringPalindrome(s);  
return 0;  
}
```

OUTPUT:

Enter a string :: ctc

ctc is a palindrome string

Enter a string :: sourav

sourav is not a palindrome string