<u>Subject Name:</u> Computer Programming with C

<u>Subject Code:</u> MCA102

<u>Assignment-2</u>

<u>Topic:</u> C Functions

<u>Name:</u> Shashwat Khaitan

<u>Section:</u> B

<u>Enrollment No:</u> 12024006015093

<u>Class Roll No:</u> 36

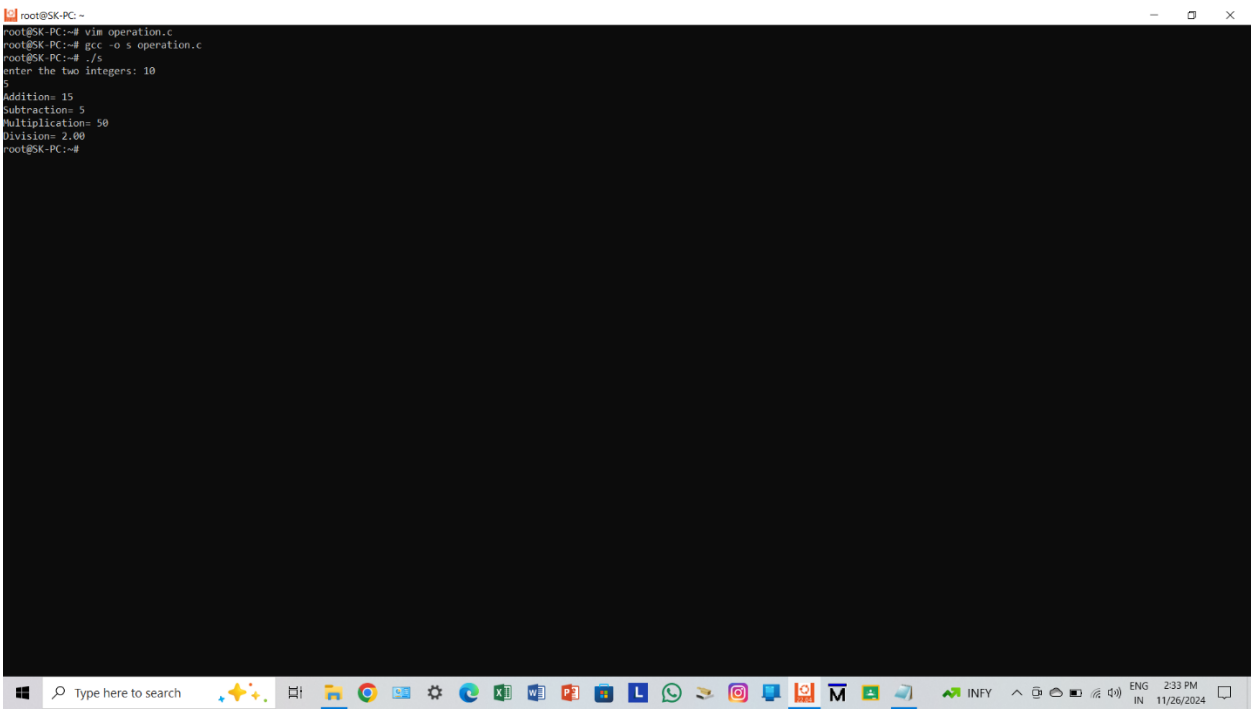
1. Write a C program to add, subtract, multiply, and divide two integers using a user-defined type function with a return type.

```
#include <stdio.h>


int add(int a, int b)

{

    return a+b;

}


int subtract(int a , int b)

{

    return a-b;
```

```
}

int multiply(int a, int b)
{
	return a*b;
}

float divide(int a, int b)
{
	return (float)a/b;
}

int main()
{
	int a,b;
	printf("enter the two integers: ");
	scanf("%d %d", &a, &b);
	printf("Addition= %d\n", add(a,b));
	printf("Subtraction= %d\n", subtract(a,b));
	printf("Multiplication= %d\n", multiply(a,b));
	if(b!=0)
```

```
        printf("Division= %.2f\n", divide(a,b));
    else
        printf("Division by zero is not allowed.\n");


    return 0;
}
```

OUTPUT

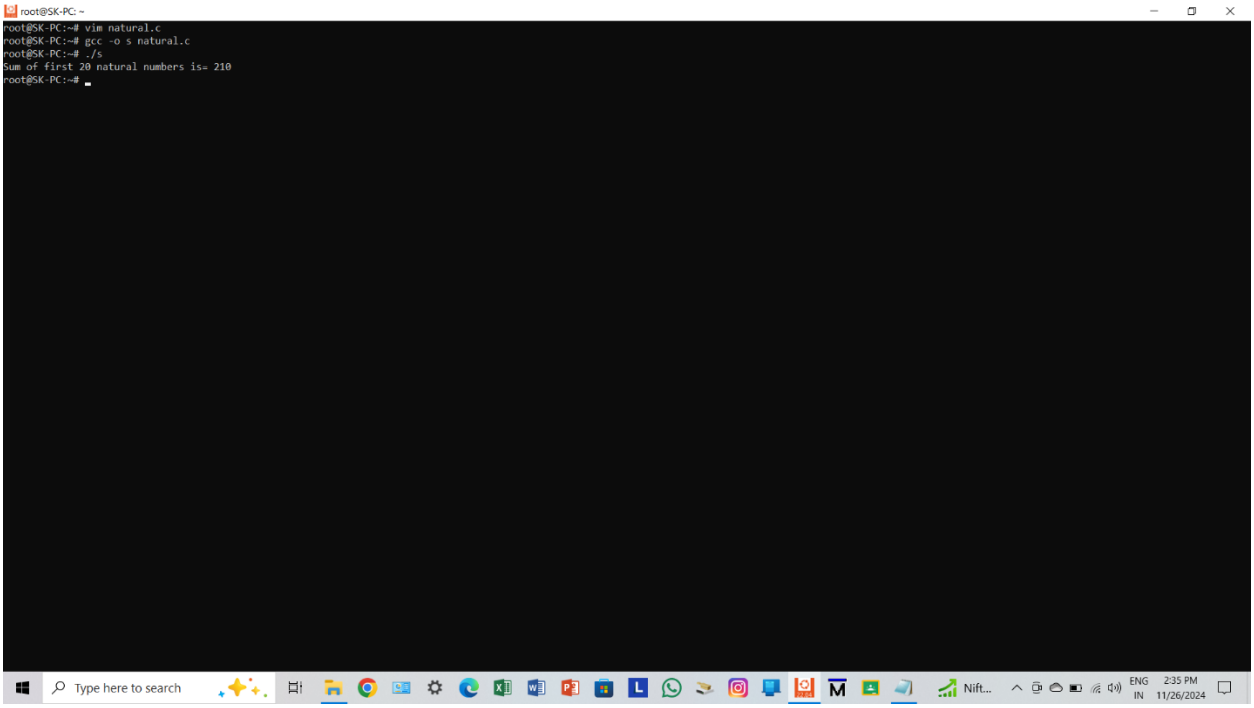


2. Write a C program to calculate the sum of the first 20 natural numbers using a recursive function.

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

```
int sum(int n)
{
    if(n==1)
        return 1;
    return n+sum(n-1);
}

int main()
{
    printf("Sum of first 20 natural numbers is= %d\n", sum(20));
    return 0;
}
```

OUTPUT

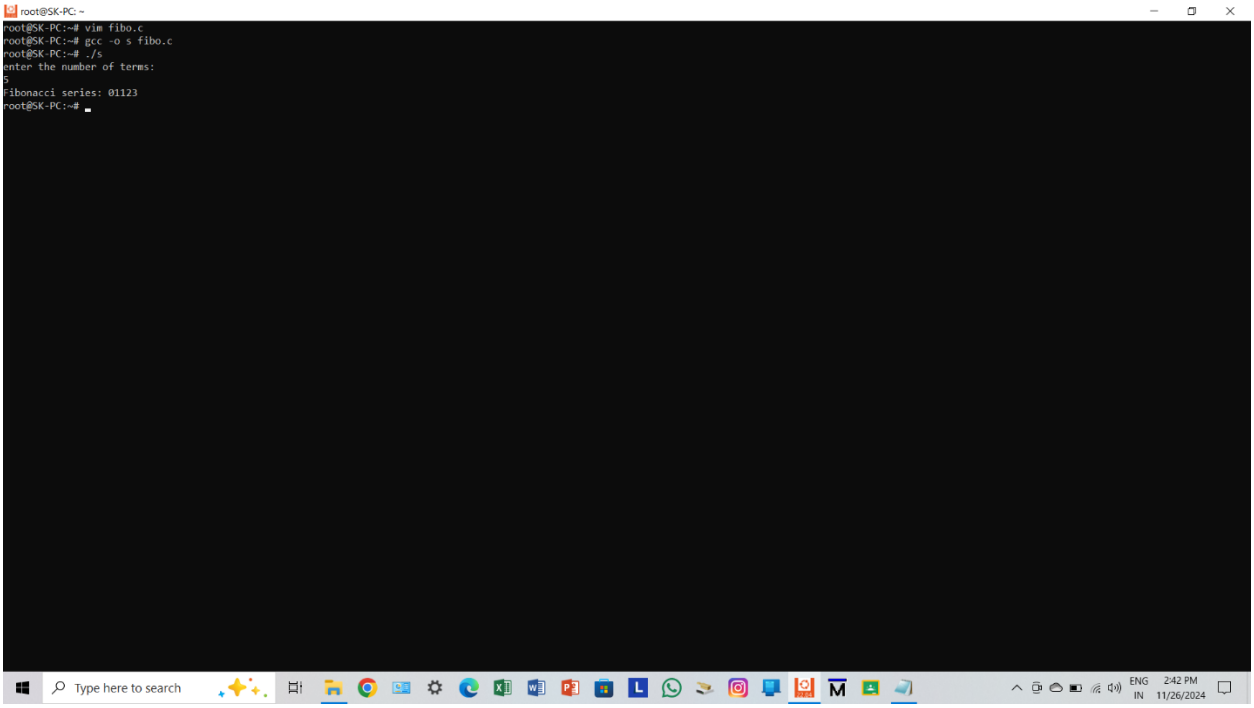


3. Write a C program to generate a Fibonacci series using a recursive function.

```
#include <stdio.h>


int fibo(int n)

{

    if(n==0)

        return 0;

    if(n==1)

        return 1;

    return fibo(n-1) + fibo(n-2);
```

```
}


int main()
{
    int n,i;
    printf("enter the number of terms: ");
    scanf("%d", &n);
    printf("Fibonacci series: ");
    for(i=0;i<n;i++)
    {
        printf("%d", fibo(i));
    }
    printf("\n");
    return 0;
}
```

OUTPUT

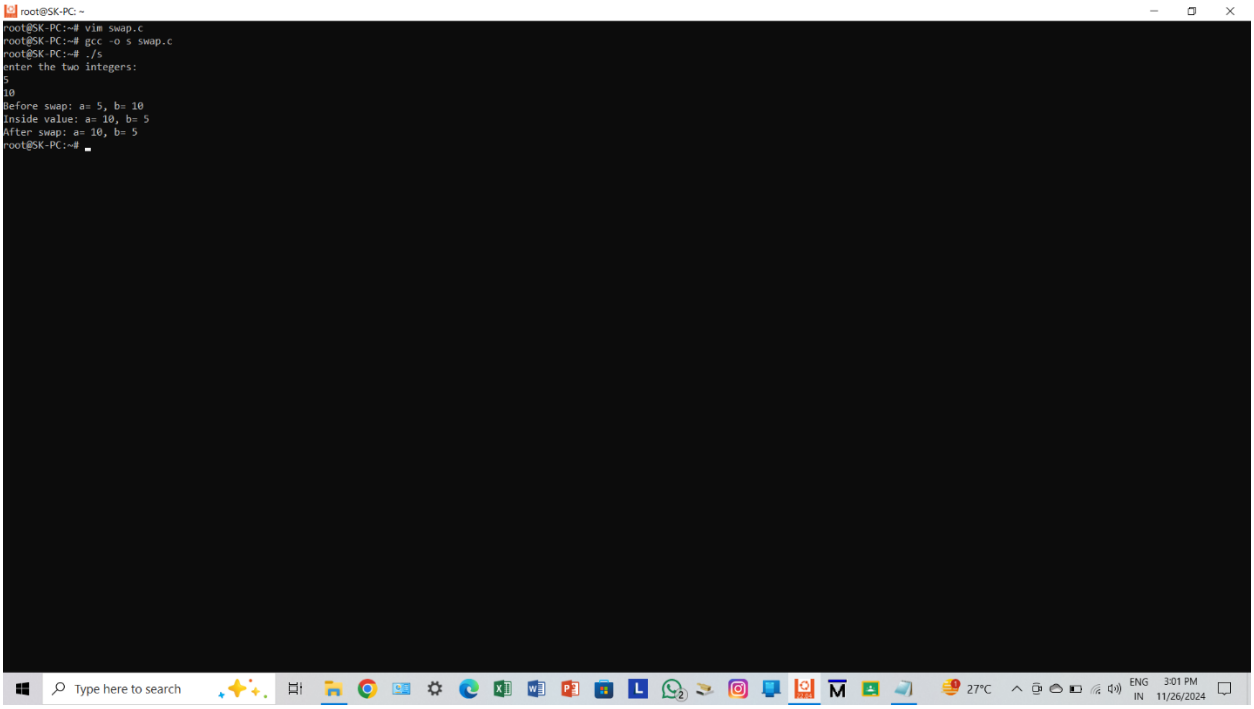


4. Write a C program to swap two integers using call-by-value and call-by-reference methods of passing arguments to a function.

```
#include <stdio.h>


void value(int a, int b)

{

    int temp=a;

    a=b;

    b=temp;

    printf("Inside value: a= %d, b= %d\n", a,b);

}
```

```
void reference(int *a, int *b)
{
	int temp= *a;
	*a= *b;
	*b= temp;
}

int main()
{
	int a,b;
	printf("enter the two integers: ");
	scanf("%d %d", &a, &b);
	printf("Before swap: a= %d, b= %d\n", a,b);
	value(a,b);
	reference(&a, &b);
	printf("After swap: a= %d, b= %d\n", a,b);

	return 0;
}
```

OUTPUT

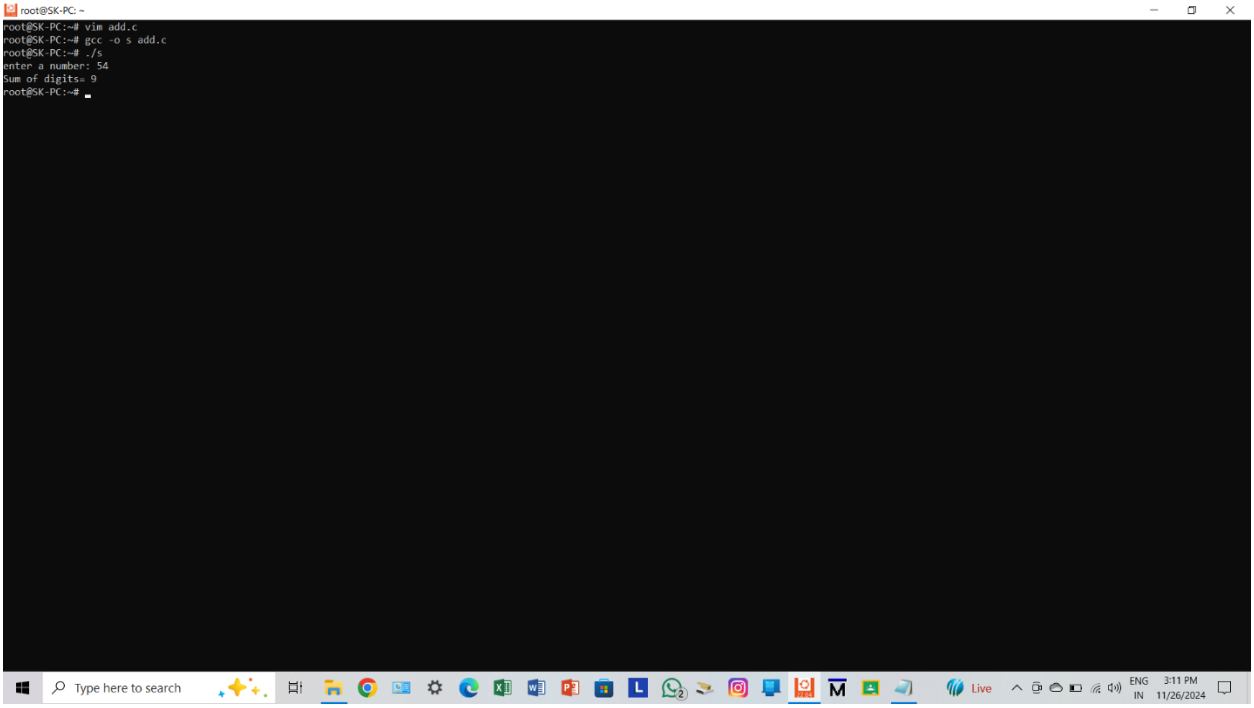


5. Write a C program to find the sum of the digits of the number using a recursive function.

```
#include <stdio.h>


int add(int n)

{

    if(n==0)

        return 0;

    return n%10+add(n/10);

}
```

```
int main()
{
    int n;
    printf("enter a number: ");
    scanf("%d", &n);
    printf("Sum of digits= %d\n", add(n));
    return 0;
}
```

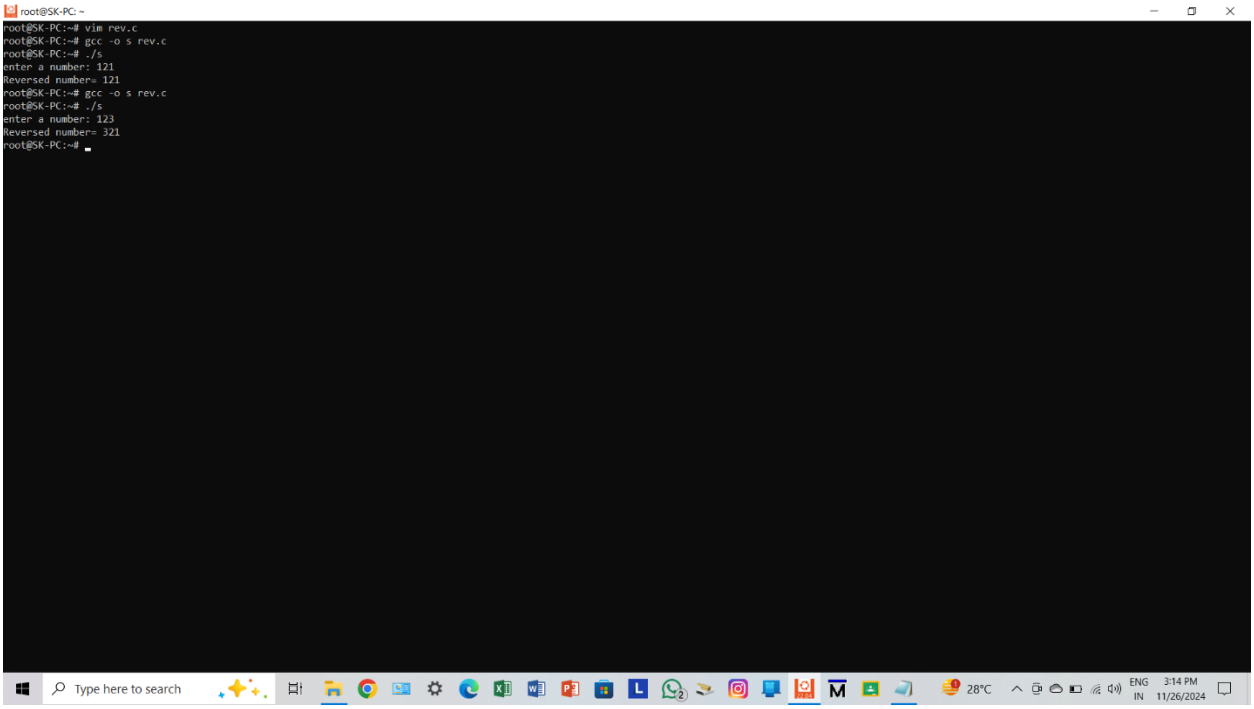
OUTPUT



6. Write a C program to read an integer number and print the reverse of that number using recursion.

```
#include <stdio.h>


int reverse(int n, int r)
{
    if(n==0)
        return r;
    return reverse(n/10, r*10+n%10);
}


int main()
{
    int n, rev;
    printf("enter a number: ");
    scanf("%d", &n);
    rev=reverse(n,0);
    printf("Reversed number= %d\n", rev);
    return 0;
}
```

OUTPUT

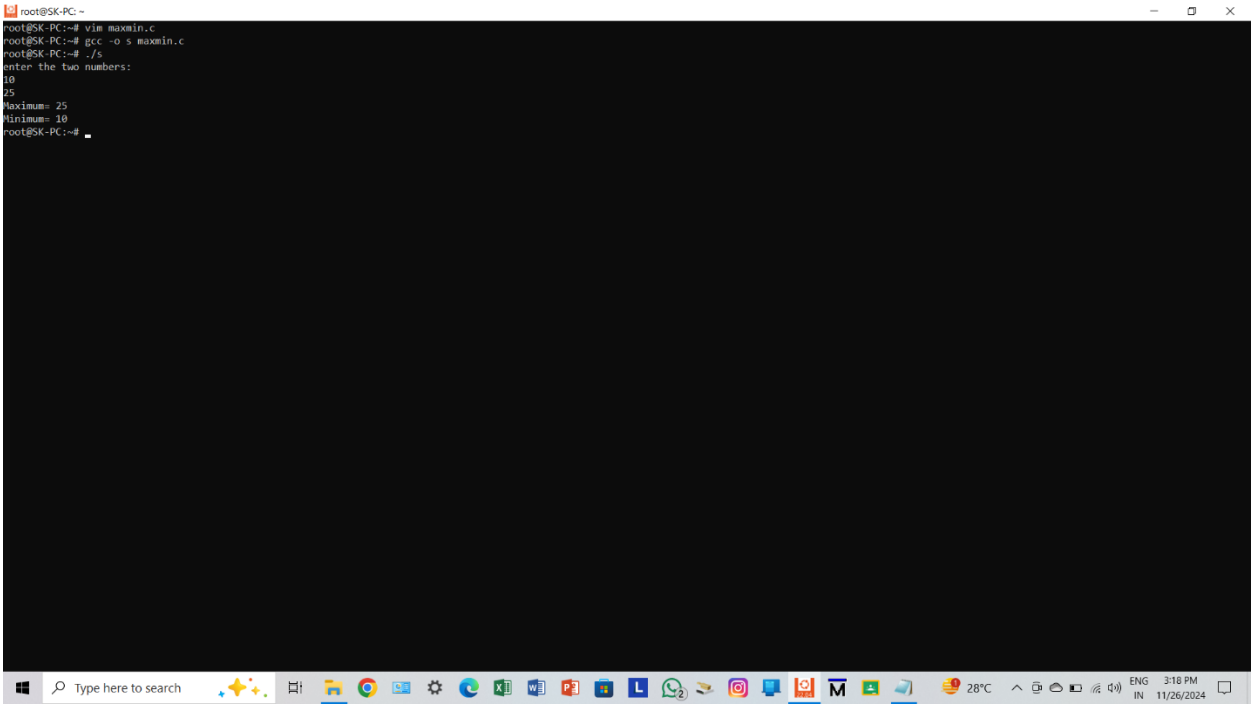


7. Using functions, write a C program to find the maximum and minimum between two numbers.

```
#include <stdio.h>


int max(int a, int b)

{

      return (a>b) ? a:b;

}


int min(int a, int b)
```

```
{
	return (a<b) ? a:b;
}

int main()
{
	int a,b;
	printf("enter the two numbers: ");
	scanf("%d %d", &a, &b);
	printf("Maximum= %d\n", max(a,b));
	printf("Minimum= %d\n", min(a,b));
	return 0;
}
```

OUTPUT

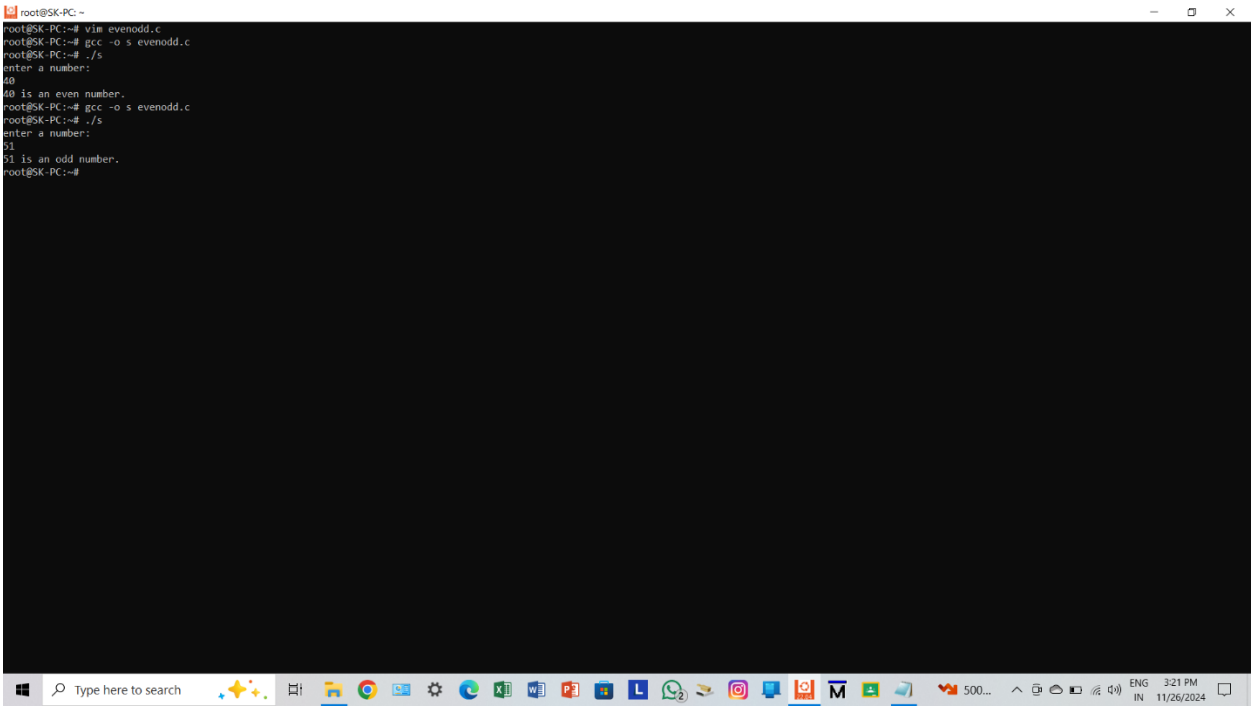


8. Write a C program to check whether a number is even or odd using functions.

```
#include <stdio.h>


int isEven(int n)

{

      return n%2==0;

}


int main()

{
```

```
    int n;
    printf("enter a number: ");
    scanf("%d", &n);
    if(isEven(n))
        printf("%d is an even number.\n", n);
    else
        printf("%d is an odd number.\n", n);

    return 0;
}
```

OUTPUT



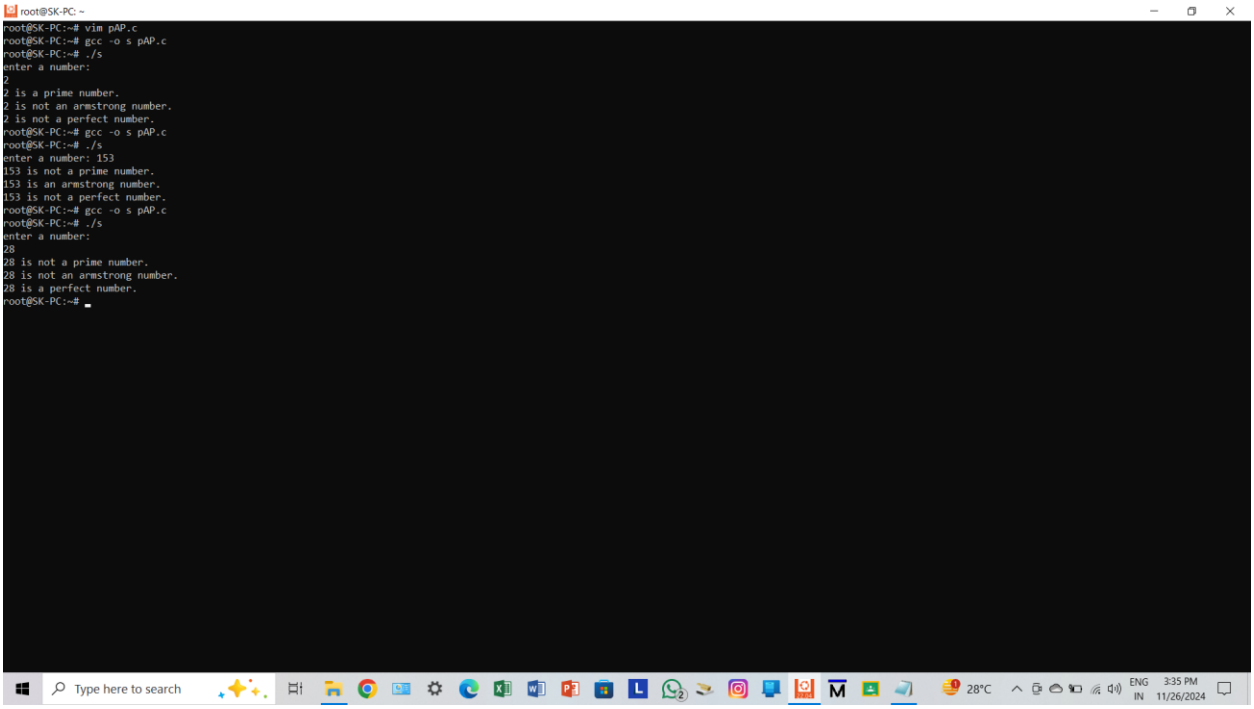
9. Write a C program to check whether a number is a prime, Armstrong, or Perfect number using functions.

```
#include <stdio.h>


int isPrime(int n)
{
    if(n<=1)
        return 0;
    for(int i=2;i*i<=n;i++)
    {
        if(n%i==0)
            return 0;
    }
    return 1;
}


int isArmstrong(int n)
{
    int s=0, temp=n, d;
    while(temp>0)
    {
```

```
        d=temp%10;
        s=s+d*d*d;
        temp=temp/10;
    }
    return s==n;
}

int isPerfect(int n)
{
    int sum=0;
    for(int i=1;i<n;i++)
    {
        if(n%i==0)
            sum=sum+i;
    }
    return sum==n;
}

int main()
{
    int n;
```

```
    printf("enter a number: ");
    scanf("%d", &n);
    if(isPrime(n))
        printf("%d is a prime number.\n", n);
    else
        printf("%d is not a prime number.\n", n);

    if(isArmstrong(n))
        printf("%d is an armstrong number.\n", n);
    else
        printf("%d is not an armstrong number.\n", n);

    if(isPerfect(n))
        printf("%d is a perfect number.\n", n);
    else
        printf("%d is not a perfect number.\n", n);

    return 0;
}
```

OUTPUT

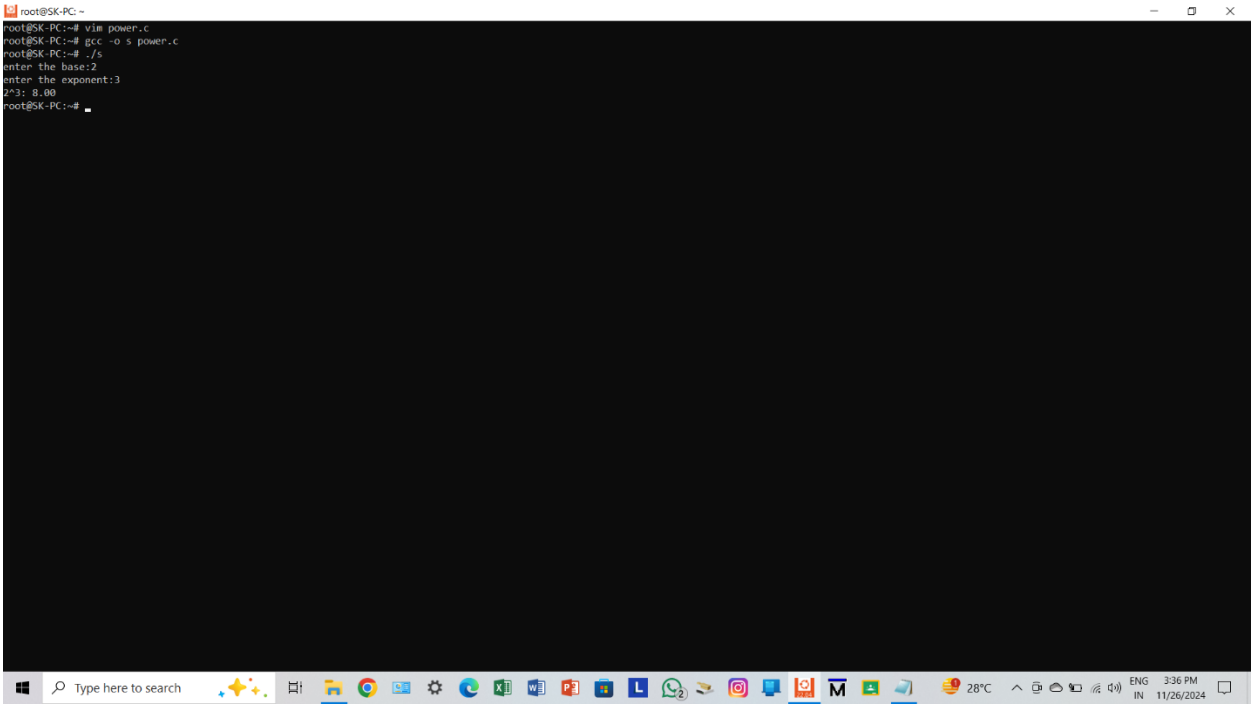


10.Write a C program to find the power of any number using recursion.

```
#include <stdio.h>
double power(int base, int exp)
{
    if(exp==0)
        return 1;
    else if(exp<0)
        return 1/power(base, -exp);
    else if(exp%2==0)
        return power(base*base, exp/2);
    else
```

```
        return base*power(base*base, (exp-1)/2);
}

int main()
{
    int base, expo;
    printf("enter the base:");
    scanf("%d", &base);
    printf("enter the exponent:");
    scanf("%d", &expo);
    printf("%d^%d: %.2f\n", base, expo, power(base,expo));
    return 0;
}
```

OUTPUT


root@SK-PC: ~
root@SK-PC:~# vim power.c
root@SK-PC:~# gcc -o s power.c
root@SK-PC:~# ./s
enter the base:2
enter the exponent:3
2^3: 8.00
root@SK-PC:~#
Type here to search
28°C
ENG
IN
3:36 PM
11/26/2024