

Subject Name: Computer Programming with C

Subject Code: MCA102

Assignment-2

Topic: C Functions

Name: Shashwat Khaitan

Section: B

Enrollment No: 12024006015093

Class Roll No: 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));
```

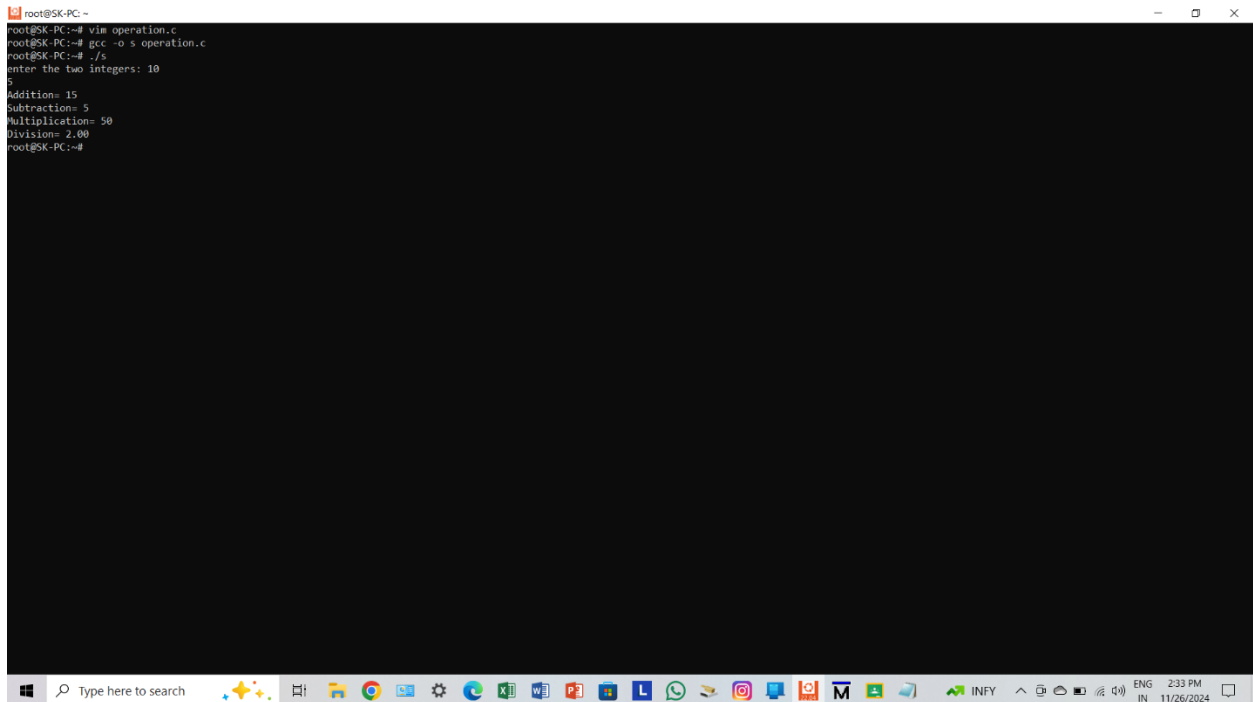
```
    printf("Subtraction= %d\n", subtract(a,b));
```

```
    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



```
root@SK-PC: ~  
root@SK-PC:~# vim operation.c  
root@SK-PC:~# gcc -o s operation.c  
root@SK-PC:~# ./s  
enter the two integers: 10  
5  
Addition= 15  
Subtraction= 5  
Multiplication= 50  
Division= 2.00  
root@SK-PC:~#
```

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

```
root@SK-PC: ~  
root@SK-PC:~# vim natural.c  
root@SK-PC:~# gcc -o s natural.c  
root@SK-PC:~# ./s  
Sum of first 20 natural numbers is= 210  
root@SK-PC:~#
```

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

```
root@SK-PC: ~  
root@SK-PC:~# vim fibo.c  
root@SK-PC:~# gcc -o s fibo.c  
root@SK-PC:~# ./s  
enter the number of terms:  
5  
Fibonacci series: 01123  
root@SK-PC:~#
```

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

```
root@SK-PC: ~  
root@SK-PC:~# vim swap.c  
root@SK-PC:~# gcc -o s swap.c  
root@SK-PC:~# ./s  
enter the two integers:  
5  
10  
Before swap: a= 5, b= 10  
Inside value: a= 10, b= 5  
After swap: a= 10, b= 5  
root@SK-PC:~#
```

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: ");

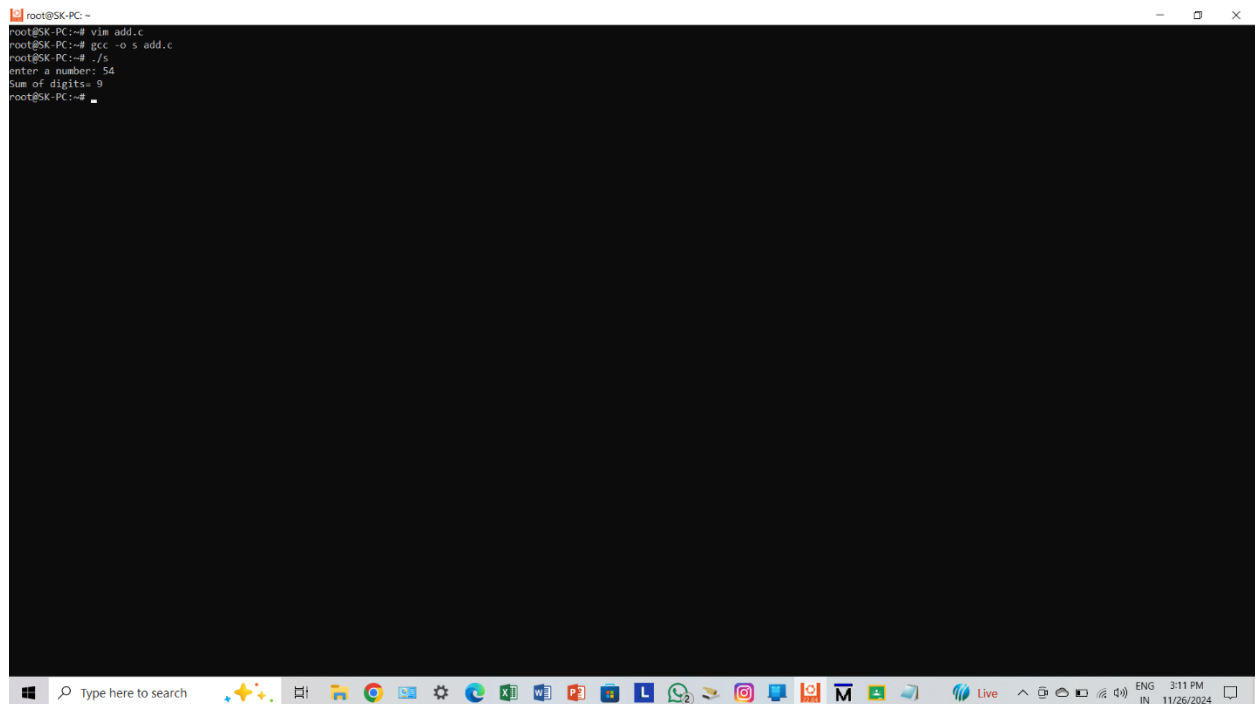
    scanf("%d", &n);

    printf("Sum of digits= %d\n", add(n));

    return 0;

}
```

OUTPUT



```
root@SK-PC: ~  
root@SK-PC:~# vim add.c  
root@SK-PC:~# gcc -o s add.c  
root@SK-PC:~# ./s  
enter a number: 54  
Sum of digits= 9  
root@SK-PC:~#
```

The screenshot shows a Windows 10 desktop environment. At the top, a terminal window titled 'root@SK-PC: ~' displays the execution of a C program. The user enters '54' in response to the prompt 'enter a number:'. The program outputs 'Sum of digits= 9'. The Windows taskbar is visible at the bottom, showing various application icons and the system clock indicating 3:11 PM on 11/26/2024.

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);
```

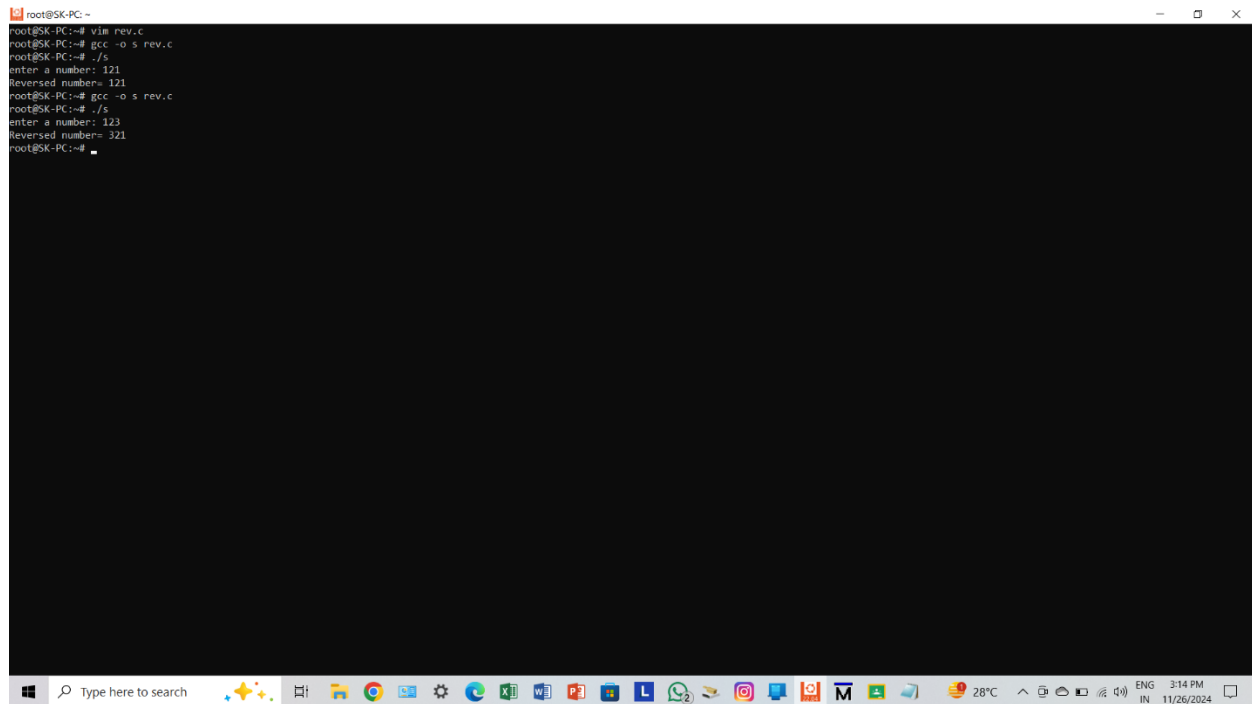
```
    printf("Reversed number= %d\n", rev);
```

```
    return 0;
```

```
}
```

OUTPUT

```
root@SK-PC: ~  
root@SK-PC:~# vim rev.c  
root@SK-PC:~# gcc -o s rev.c  
root@SK-PC:~# ./s  
enter a number: 121  
Reversed number= 121  
root@SK-PC:~# gcc -o s rev.c  
root@SK-PC:~# ./s  
enter a number: 123  
Reversed number= 321  
root@SK-PC:~#
```



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

```
root@SK-PC: ~  
root@SK-PC:~# vim maxmin.c  
root@SK-PC:~# gcc -o s maxmin.c  
root@SK-PC:~# ./s  
enter the two numbers:  
10  
25  
Maximum= 25  
Minimum= 10  
root@SK-PC:~#
```

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);

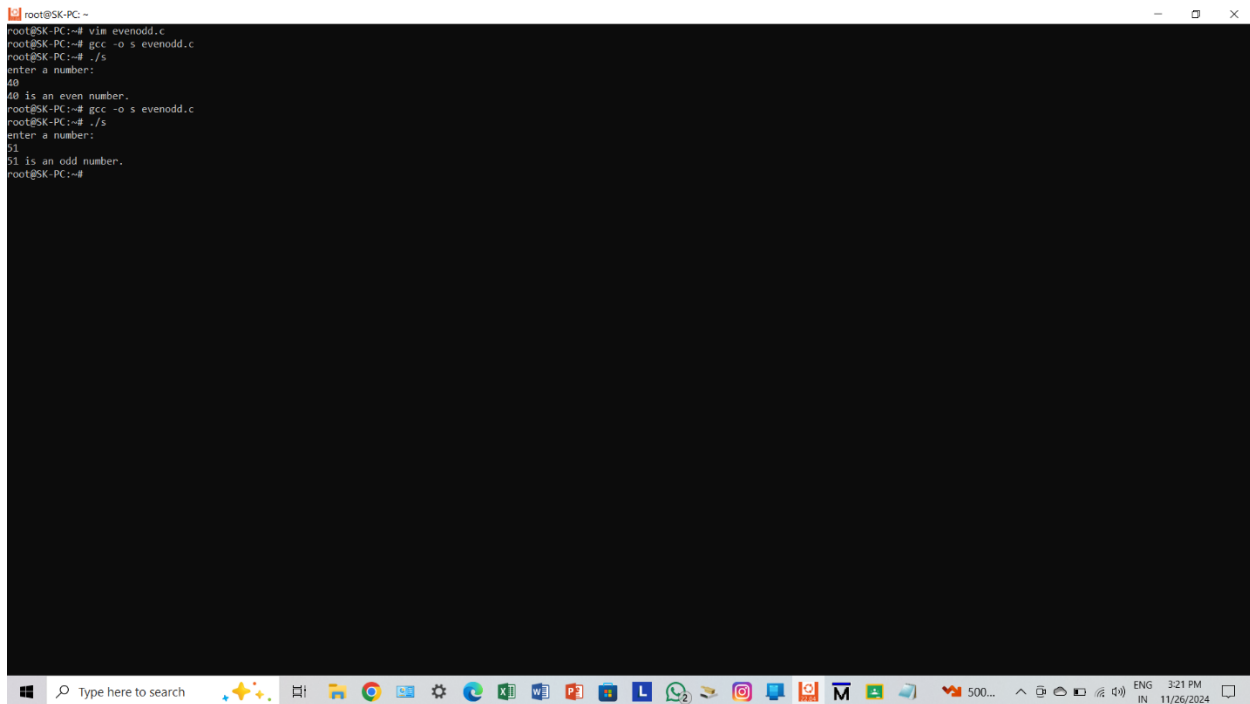
else

    printf("%d is an odd number.\n", n);

return 0;

}
```

OUTPUT



```
root@SK-PC:~# vim evenodd.c
root@SK-PC:~# gcc -o s evenodd.c
root@SK-PC:~# ./s
enter a number:
40
40 is an even number.
root@SK-PC:~# gcc -o s evenodd.c
root@SK-PC:~# ./s
enter a number:
51
51 is an odd number.
root@SK-PC:~#
```

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;
}
```

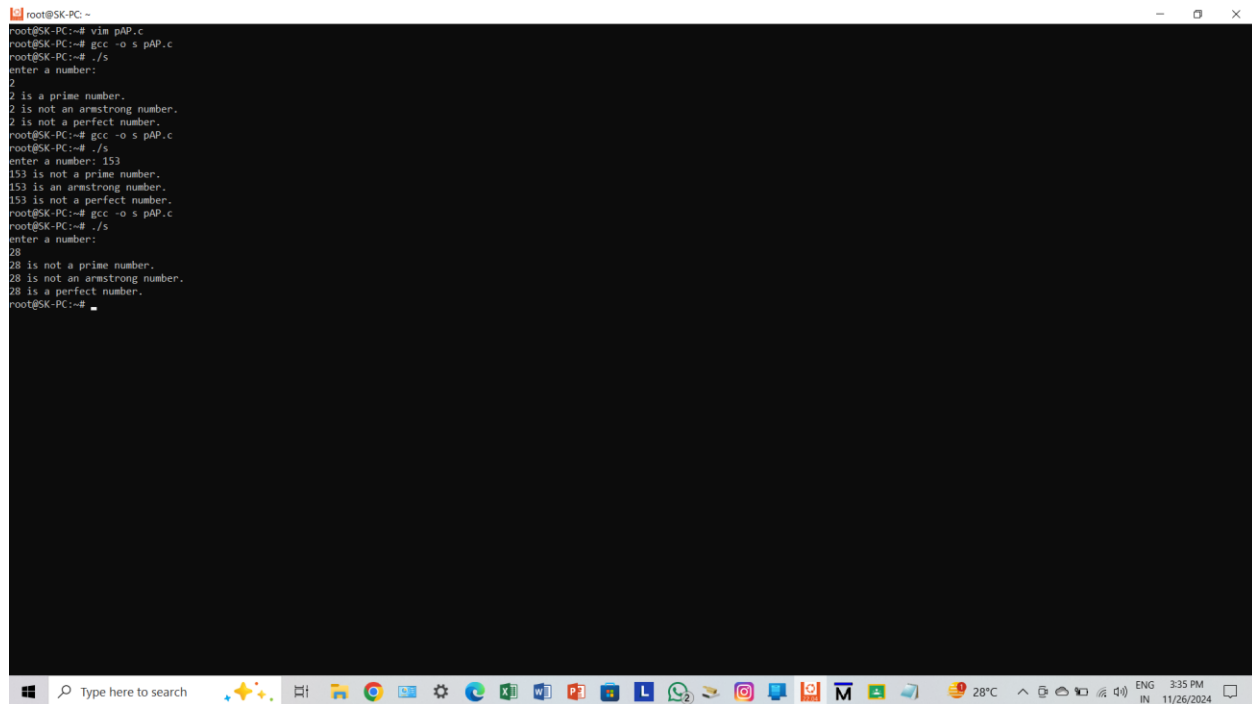
```
int isPerfect(int n)
{
    int sum=0;
    for(int i=1;i<n;i++)
    {
        if(n%i==0)
            sum=sum+i;
    }

    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



```
root@SK-PC: ~  
root@SK-PC:~# vim pAP.c  
root@SK-PC:~# gcc -o s pAP.c  
root@SK-PC:~# ./s  
enter a number:  
2  
2 is a prime number.  
2 is not an armstrong number.  
2 is not a perfect number.  
root@SK-PC:~# gcc -o s pAP.c  
root@SK-PC:~# ./s  
enter a number: 153  
153 is not a prime number.  
153 is an armstrong number.  
153 is not a perfect number.  
root@SK-PC:~# gcc -o s pAP.c  
root@SK-PC:~# ./s  
enter a number:  
28  
28 is not a prime number.  
28 is not an armstrong number.  
28 is a perfect number.  
root@SK-PC:~#
```

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 5 power.c  
root@SK-PC:~# ./5  
enter the base:2  
enter the exponent:3  
2^3: 8.00  
root@SK-PC:~#
```

The image shows a Windows taskbar at the bottom of the terminal window. It includes a search bar on the left, followed by a row of application icons: File Explorer, Google Chrome, Microsoft Edge, Settings, Microsoft Word, Microsoft Excel, Microsoft PowerPoint, Outlook, WhatsApp, Telegram, Instagram, and a folder icon. On the right side of the taskbar, there is a system tray containing a temperature indicator showing 28°C, a volume icon, a network icon, and a clock displaying 3:36 PM on 11/26/2024.