WeekNo	PPA	PROGRAM
MOCK	1	
	2	
	3	
	4	
	5	
	6	

MOCK – T2_2025 - Program 1

AIM : Write a C program that reads the coordinates of a point x and y as integers and determines in which quadrant the point lies. Assume that neither of the coordinates will be 0.

- If both x and y are positive, print Quadrant 1.
- If x is negative and y is positive, print Quadrant 2.
- If both x and y are negative, print Quadrant 3.
- If x is positive and y is negative, print Quadrant 4.

Input Format
Two integers x and y are separated by a space.

Output Format
Print the quadrant in which the point lies

Sample Input
5 -7
Output
Quadrant 4

PROGRAM :

```
#include <stdio.h>
int main() {
    int p,q;
   scanf("%d %d",&p,&q);
   if((p==0) && (q==0))
     printf("Origin");
   else if(p==0)
     printf("y-axis");
   else if(q==0)
     printf("x-axis");
   else if((p>0)&&(q>0))
     printf("Quadrant 1");
   else if((p<0)&&(q>0))
     printf("Quadrant 2");
   else if((p<0)&&(q<0))
     printf("Quadrant 3");
```

```
   else
     printf("Quadrant 4");
return 0;
}
```

OUTPUT:

Sample Input1

```
5 -7
```

Output

```
Quadrant 4
```

Sample Input2

4 4

Output

```
Quadrant 1
```

Sample Input3

-4 -4

Output

```
Quadrant 3
```

Sample Input4

-4 4

Output

```
Quadrant 2
```

MOCK – T2_2025 - Program 2

AIM : Write a C program that prints a discount based on a customer's membership level and their purchase amount. The discount rules are as follows:

- **Gold Member(G):**
  - Purchase >= 1000: Discount 20%
  - Purchase < 1000: Discount 15%
- **Silver Member(S):**
  - Purchase >= 1000: Discount 10%
  - Purchase < 1000: Discount 5%
- **Regular Customer(R):**
  - Purchase >= 1000: Discount 3%
  - Purchase < 1000: Discount 1%

Input Format:
First Line: A character (G/S/R) represent the membership level

Second line: An Integer represents the purchase amount

Output Format:
Discount amount as float with two decimal point

PROGRAM:

```
#include<stdio.h>
int main()
{
   float pamt,dis,damt;
   char mlevel;
   scanf("%c",&mlevel);
   scanf("%f",&pamt);
   switch(mlevel)
   {
      case 'G': if(pamt<1000)
                  dis=15;
                else
                  dis=20;
                break;
      case 'S': if(pamt<1000)
                  dis=5;
                else
```

```
            dis=10;
          break;
      case 'R': if(pamt<1000)
            dis=1;
          else
            dis=3;
          break;

   }
   damt=(pamt*dis)/100.0;
   printf("%.2f",damt);
   return 0;
}
```

OUTPUT:

Sample Input 1

```
G
1200
```

Output

```
240.00
```

Sample Input 2

```
S
950
```

Output

```
47.50
```

Sample Input 3

```
G
2500
```

Output

```
500.00
```

Sample Input 4

```
R
1000
```

Output

```
30.00
```

MOCK – T2_2025 - Program 3

PROGRAM 1

AIM : Write a function called **total** that takes three positive integers: start, end, and k as parameters, where start < end. The function should calculate and return the sum of all even numbers within the inclusive range from start to end that are divisible by k.

Sample Input

2 //start
14 //end
3 //k

Output

18

Explanation:

Here, 2, 4, 6, 8, 10, 12, and 14 are even numbers in between 2 and 14 and out of these, 6 and 12 are divisible by 3. Hence, output will be 6 + 12 = 18.

PROGRAM

```
#include<stdio.h>
int total(int start, int end, int k)
{
      int i,sum=0;
   if(start%2==0)
     i=start;
   else
     i=start+1;

    for(; i<=end;i+=2)
    {
       if(i%k==0)
         sum=sum+i;
    }
   return sum;
}
int main()
{
   int start, end, k;
   scanf("%d",&start);
   scanf("%d",&end);
   scanf("%d",&k);
   printf("%d",total(start, end, k));
```

```
    return 0;
}
```

OUTPUT :

Sample Input1
1
20
4

Output
60

Sample Input2
1
20
5
Output
30

MOCK – T2_2025 - Program 4

PROGRAM 1

AIM : Write a function **print_pattern** that takes an integer n and prints the following type of pattern:
Let n = 5.
Output
1
2 1
3 2 1
4 3 2 1
5 4 3 2 1
Note:- There is no space after last digit in each line.

Sample Input
3
Output
1
2 1
3 2 1

PROGRAM :

```
#include<stdio.h>
void print_pattern(int n)
{
      int l,j;
for(l=1;l<=n;l++)
{
   for(j=l;j>=2;j--)
      printf("%d ",j);
   printf("1");
   printf("\n");
}

}
int main()
```

```
{
   int n;
   scanf("%d",&n);
   print_pattern(n);
   return 0;
}
```

OUTPUT :

Sample Input1

3

Output

```
1
2 1
3 2 1
```

Sample Input2

5

Output

```
1
2 1
3 2 1
4 3 2 1
5 4 3 2 1
```

MOCK – T2_2025 - Program 5

PROGRAM 1

AIM : An **Armstrong number** is an n-digit number if the sum of its digits, each raised to the power of n, is equal to the original number itself.
For example, let's consider a 3-digit number:

In this case, 153 is an Armstrong number because the sum of the cubes of its digits equals the number itself (153).
Write a function **countArmstrong** that takes three integer pointers a, b, and count as parameters. The function counts all Armstrong numbers in between the value at a and b(both inclusive) and stores the count at the pointer count.
Note:- The function does not return anything. It stores the count of the Armstrong numbers at pointer count. Both values at pointer a and b are positive integers and a < b.
Sample Input
1 155
Output
10
Explanation
1, 2, 3, 4, 5, 6, 7, 8, 9 and 153 are the Armstrong numbers in the range of 1 to 155.

PROGRAM :

```
#include <stdio.h>
#include <math.h>
// Write solution code below

int armstrong(int n)
{
 long int result = 0, power=1;
 int cnt=0,r,num;
   num = n;

   while (n != 0) {
      n /= 10;
      ++cnt;
   }

   n = num;
```

```
    while (n != 0) {
        r = n % 10;
        power=pow(r,cnt);
        result += power;
        n /= 10;
    }

    if (result == num)
        return 1;
    else
        return 0;

}


void countArmstrong(int *a,int *b,int *count)
{
    int i;
    *count=0;
    for(i=*a; i<=*b;i++)
    {
      if(armstrong(i))
       *count=*count+1;
    }
}

int main()
{
    int a,b,count;
    scanf("%d %d",&a,&b);
    countArmstrong(&a,&b,&count);
    printf("%d",count);
    return 0;
}
```

OUTPUT :

Sample Input1
1 10
Output
9

Sample Input2

1 155
Output
10

Sample Input3
1 10000
Output
16

Sample Input4
10 10000
Output
7

MOCK – T2_2025 - Program 6

PROGRAM 1

AIM : Write a function **BinaryToDecimal** that takes an long integer number B, which represents binary number (consists of 0s and 1s only) and an **integer pointer** D as parameters. The function converts the given binary number B to a decimal number and stores the equivalent decimal number at the pointer D.

Note:- The function does not return anything, it only stores the equivalent decimal number at pointer D.

Sample Input

1111

Output

15

Explanation

Here, the value of variable B is 1111. Consider this value of B as a binary number and convert it into an equivalent decimal number. Hence, the equivalent decimal number of the binary number 1111 is 15.

Note:- You can use the given **power** function given in prefix code to return in your solution code (if required).

PROGRAM :

```
#include<stdio.h>
int power(int x,int y)
{
   int pow = 1;
   for (int j = 1; j <= y;j++)
   {
      pow *= x;
   }
   return pow;
}
```

```
void BinaryToDecimal(long int B, int *D)
{
  int i=0,r;
   *D=0;
   while (B != 0) {
      r = B % 10;
      *D += r * power(2, i);
      B /= 10;
      ++i;
   }




}

int main()
{
   long int B;
   int D;
   scanf("%ld",&B);
   BinaryToDecimal(B, &D);
   printf("%d", D);
   return 0;
}
```

OUTPUT :

Sample Input1
1000

Output
8

Sample Input2
1111

Output
15

Sample Input3
111111

Output
63

Sample Input4
1

Output
1