**C PROGRAMMING PRACTICALS**

**b.Conditional statement**

1. Leap Year
2. Grading students based on marks
3. Finding salary of employee by ta da

**c.loop statement**

1. Sum of natural number n
2. Factorial of number n
3. Fibonacci series
4. Sum of digits and reverse number
5. Prime number
6. Sum of even numbers till n
7. Displaying star pattern
8. Switch statement in displaying week days by number
9. Check whether number is Armstrong or not
10. Find factorial of all numbers in range 1 to n
11. generate sum 1/fact(1) +2/fact(2) +3/fact(3)..... n/fact
12. ternary operators
13. switch statement

***PRACTICAL 2. Conditional statement***

***11***

***AIM: To find whether a year is leap year or not.***

***ALGORITHM:***

1. *Start*
2. *Declare a as integer*
3. *If a is divisible by 4 then go to (iv)*

*Else it is not a leap year*

1. *if a is divisible by 100 then go to (v)*

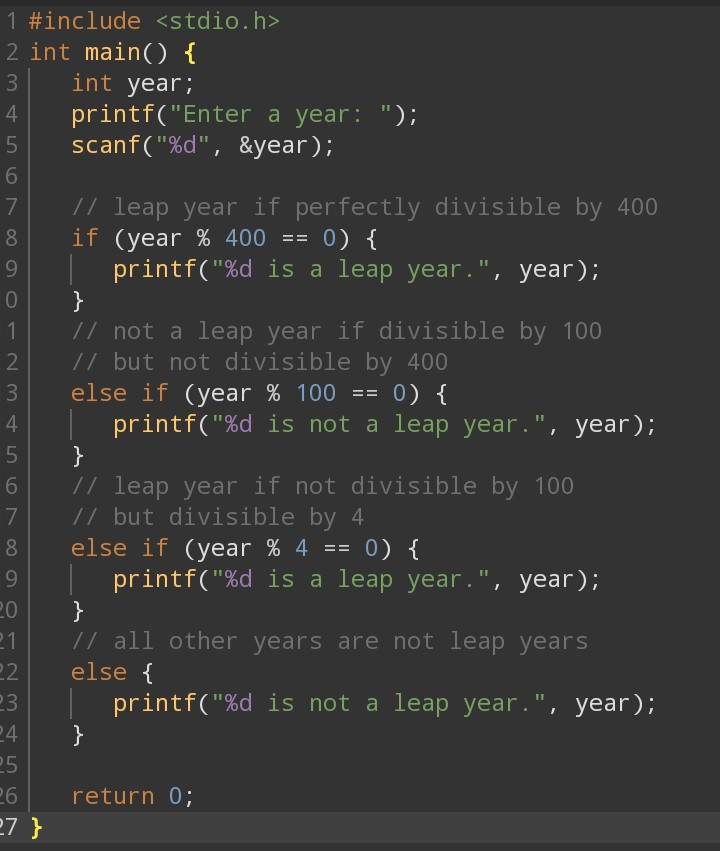
*Else it is a leap year*

1. *id a is divisible by 400 then display it’s a leap year*

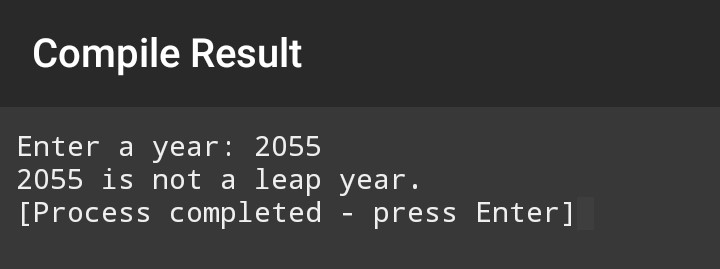
*Else display it is not a leap year*

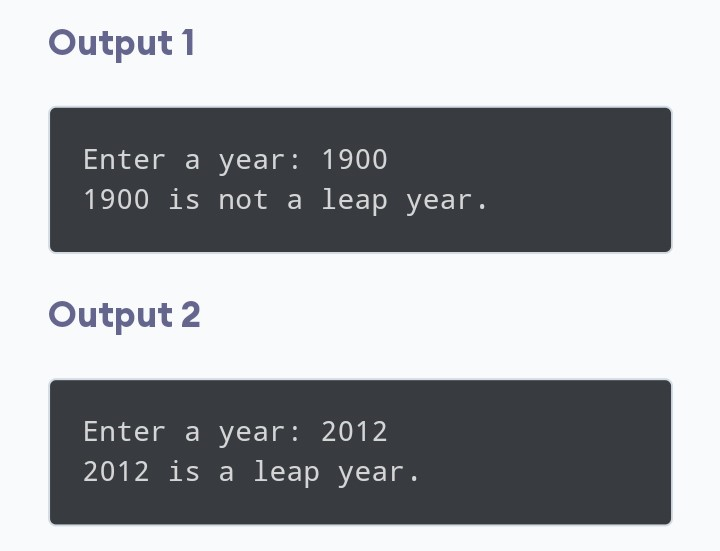
1. *Stop*

***CODING:***

**

***OUTPUT :***

**

**

***12***

**AIM: to write a program to grade a student based on marks**

**30<=marks is a c grade**

**30<=marks <70 is B grade**

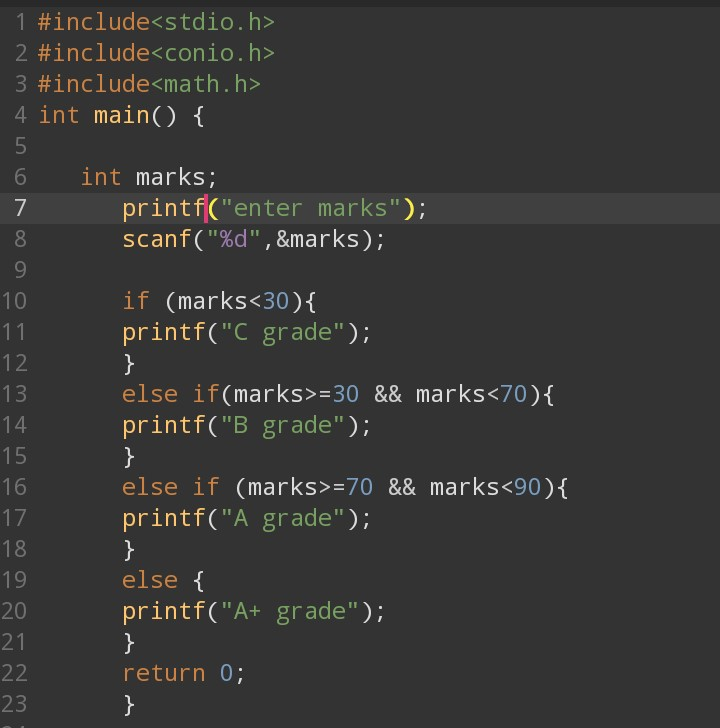
**70<=marks <90 is A grade**

**90<=marks <= is A+ grade**

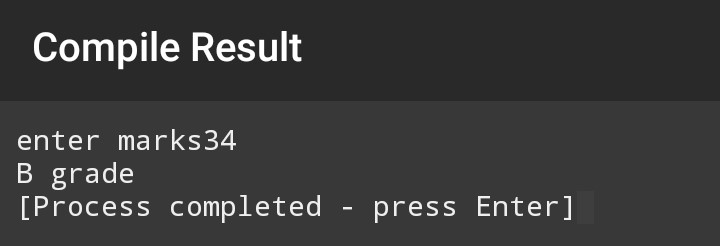
**ALGORITHM:**1. start

1. Declare marks as integer
2. If 30<= display marks is a c grade
3. Else if 30<=marks <70 display marks is B grade
4. Else if 70<=marks <90 display marks is A grade
5. Else if 90<=marks display is A+ grade

**CODING**



**OUTPUT**



**13**

**Aim : to calculate the salary of employee based on ta da**

**ALGORITHM:**

**1.start**

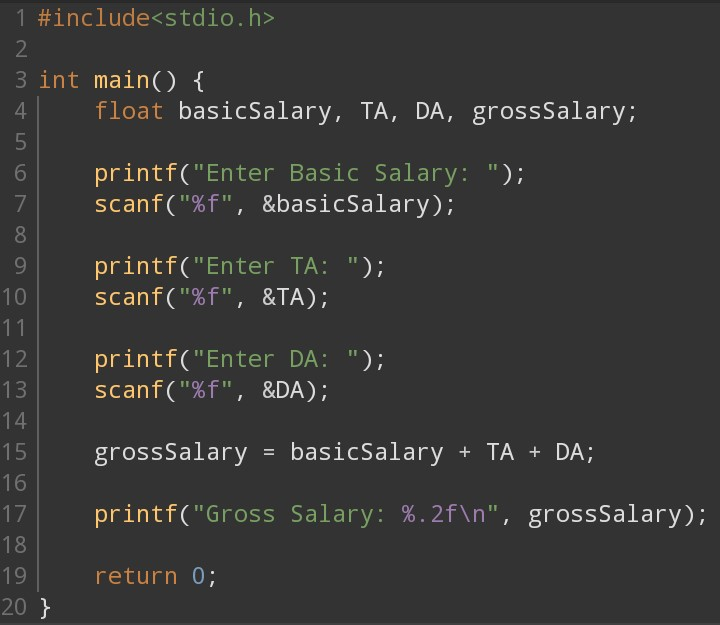
**2.declare basic salary,TA,DA,gross salary as float**

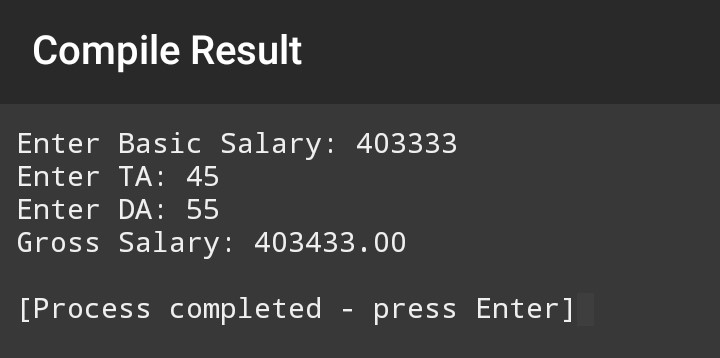
**3.input basic salary,TA,DA**

**4.gross salary=basic salary+TA+DA**

**5.display gross salary**

**6.stop**

**CODING:**

**OUTPUT:**

**PRACTICAL.3 LOOP STATEMENT**

**14. AIM : to find sum of n natural numbers**

**ALGORITHM:**

1.start

2.Declare i,n,sum as integer

3.Accept n

4.sum=0 i=1

5.Repeat following till i<=n

sum=sum+i

i=i+1

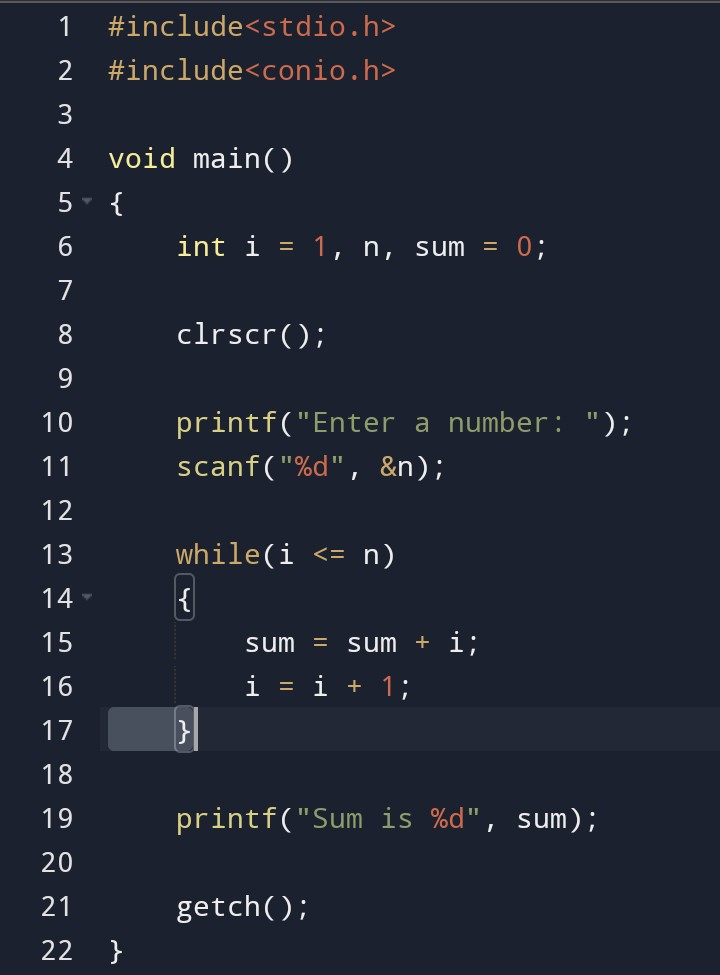
6.Display sum

7.Stop

DRYRUN  
i sum

1. 1
2. 3
3. 6
4. 10

**CODING**:



**OUTPUT**



**15.**

**AIM :Factorial of number n**

**ALGORITHM:**

1.Start

2.Declare i,n,sum as integer

3.Accept n

4.fact=1 i=1

5.Repeat following till i<=n

fact=fact\*i

i=i+1

6.Display fact

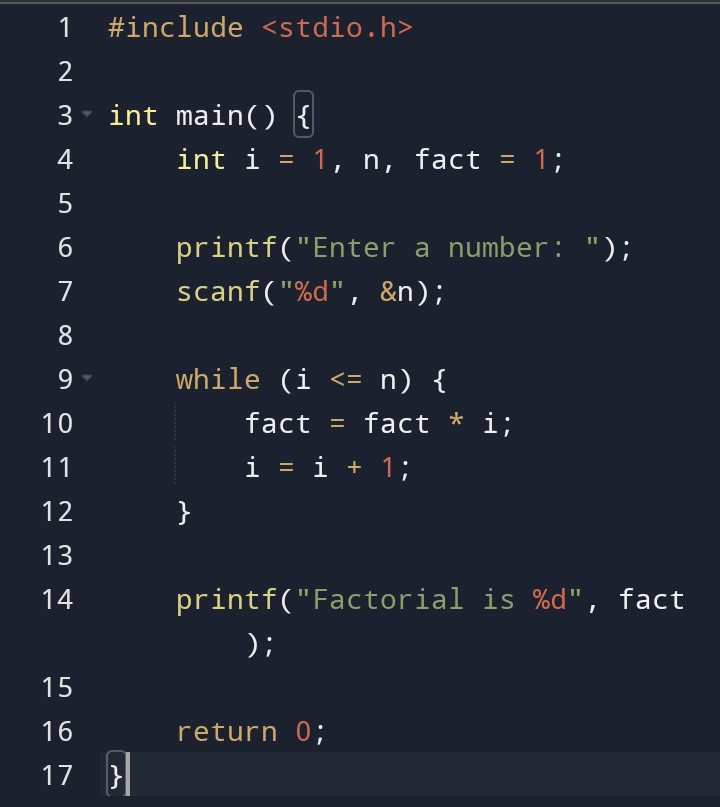
7.Stop

**Dry ryn**

**i fact**

1. **1**
2. 2
3. 6
4. 24

**CODING:**



**OUTPUT**



**16.**

**AIM:Fibonacci series**

**ALGORITHM:**

1.Start

2.Declare a=0,b=1,c,i=1,count as integer

3.Accept count

4.Display a,b

5.Repeat steps till i<=count

c=a+b;

a=b;

b=c;

i=i+1

6.Display c

7.stop

DRY RUN

A b c i

0 1 1 1

1 1 2 2

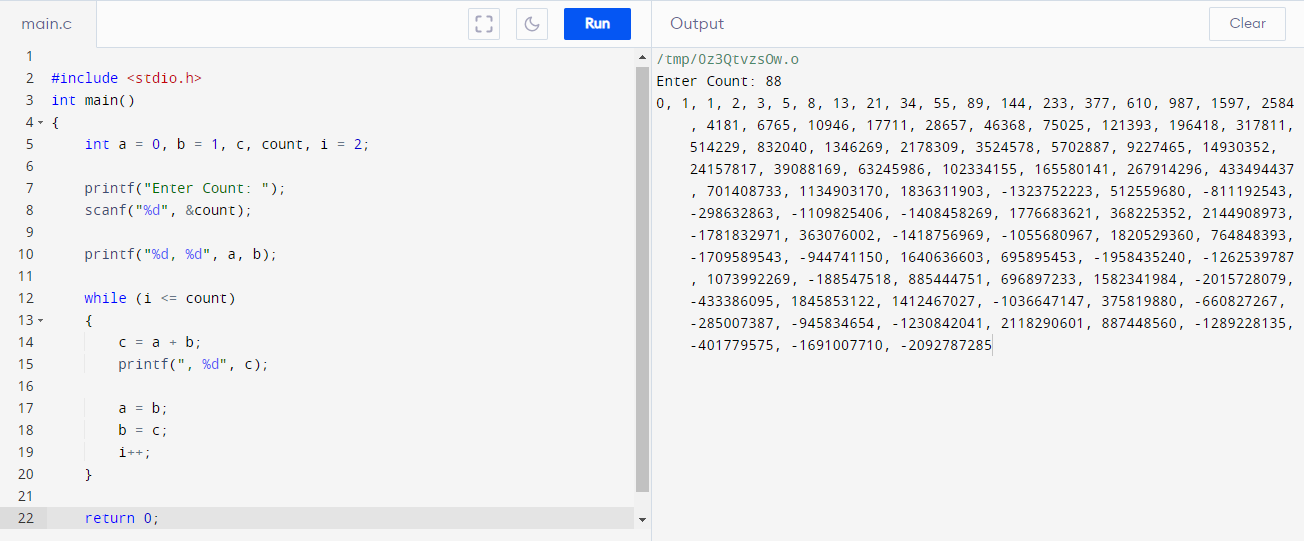
1 2 3 3

2 3 5 4

3 5 8 5

5 8 13 6 exit

**CODING:**



**17.**

**AIM:Sum of digits and reverse number**

**ALGORITHM:**

1.Start

2.Declare sum=0,n,rev=0,rem

3.Accept n

4.Repeat the following till n>0

rem=n mod10

n=n/10

sum=sum+rem

rev=rev\*10+rem

5.Display sum and rev

6.Stop

DRYRUN:

Rem n sum rev

3 123 0 0

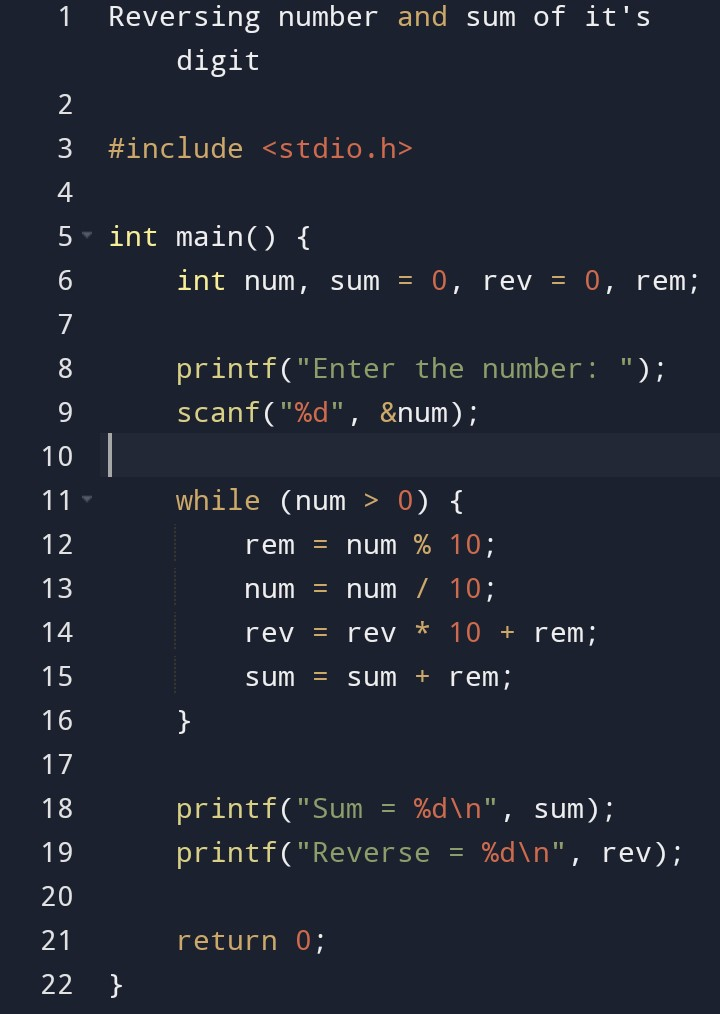
3 12 3 3

2 1 5 32

1 0 6 321

exit

**CODING:**



**OUTPUT**



**18.**

**AIM:Prime number**

**ALGORITHM:**

1)START

2)Declare int,n,i=2,flag=0

3)Input number

4)If number (n==0 || n==1)

flag=1

5)repeat the following till i<=n/2

     if n mod i==0

     then flag=1

             Break

 6)If flag=0 print “number is a prime number”

   else print”not a prime number”

7)Stop

Dry run

i flag

1 1

2 1

3 1

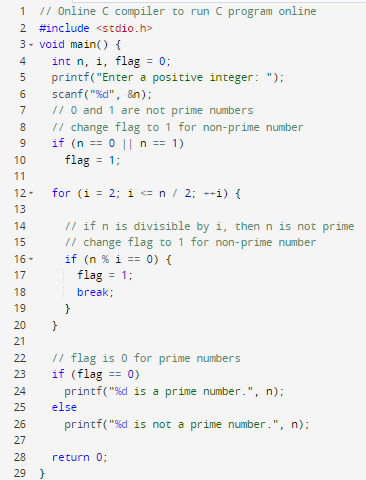
4 1

5 1

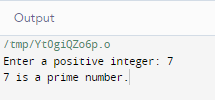
6 1

7 0

**CODING**

****

**OUTPUT**

****

**19.**

**AIM:to find sum of even numbers till n**

**ALGORITHM:**

1.start

2.Declare i,n,sum as integer

3.Accept n

4.sum=0 i=1

5.Repeat following till i<=n

sum=sum+i

i=i+1

6.Display sum

7.Stop

DRY RUN:

I sum

4 1 0

5.a 1 1

5.b 2 1

5.a 2 3

5.b 3 3

5.a 3 5

5.b 4 5

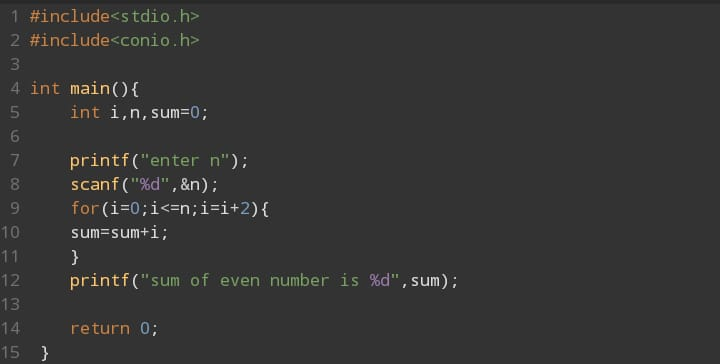
5.a 4 9

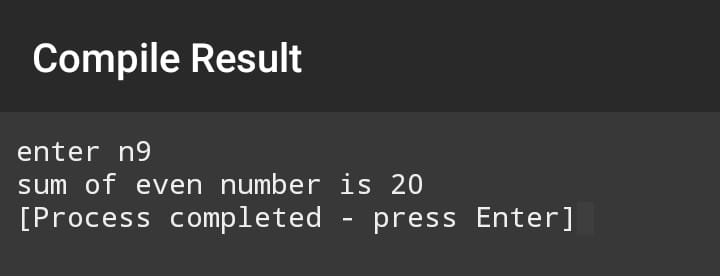
5.b 5 9

5.a 5 14

5.b 6 exit

CODING:



OUTPUT:  


**20.**

**AIM: sum of prime numbers till n**

**ALGORITHM:**

1. Initialize a variable 'sum' to 0.

2. Iterate through each number from 2 to 'n' using a for loop.

3. For each number, check if it is a prime number.

- To check if a number 'num' is prime, iterate from 2 to the square root of 'num'.

- If 'num' is divisible by any number in this range, it is not prime. Otherwise, it is prime. 4. If a number is prime, add it to the 'sum' variable.

5. After the loop ends, the 'sum' variable will hold the sum of prime numbers till 'n'. 6. Print the value of 'sum'.

DRYRUN

i flag sum

2 1 0

3 1 0

4 1 0

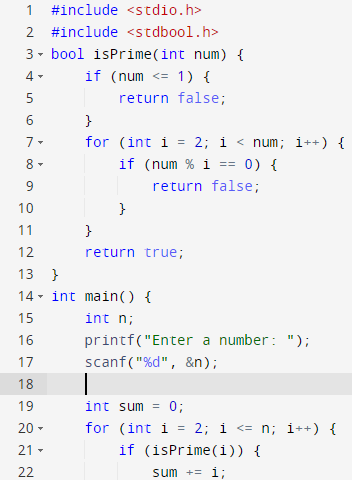
5 1 0

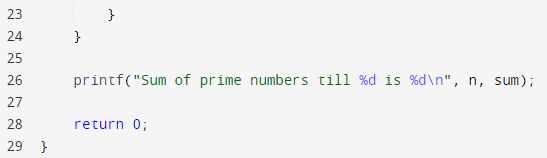
6 1 0

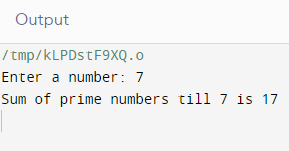
7 0 7

8 1 0

**Coding**

****

****

****

**21**

**AIM: to check whether number is Armstrong or not**

**ALGORITHM:**

1. Declare variables: num, originalNum, remainder, result, and n.

2. Initialize result as 0.

3. Take input for num.

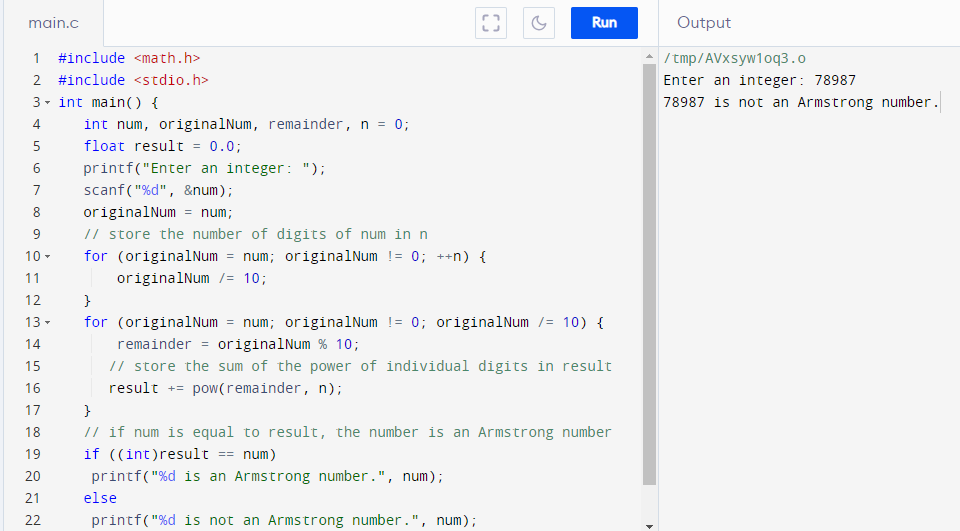
4. Set originalNum = num.

5. Find the number of digits in num and store it in n.

6. Use a while loop to iterate until num is not equal to 0. - Calculate the remainder by taking num % 10. - Raise the remainder to the power of n and add it to result. - Divide num by 10.

7. If the originalNum is equal to result, print "Armstrong number." Otherwise, print "Not an Armstrong number."

CODING:



**22**

**AIM: to find factorial of all numbers from range 1 to n**

**ALGORITHM:**

1. Declare variables: n, i, and factorial.

2. Take input for n.

3. Initialize factorial as 1.

4. Use a for loop to iterate from i = 1 to i <= n.

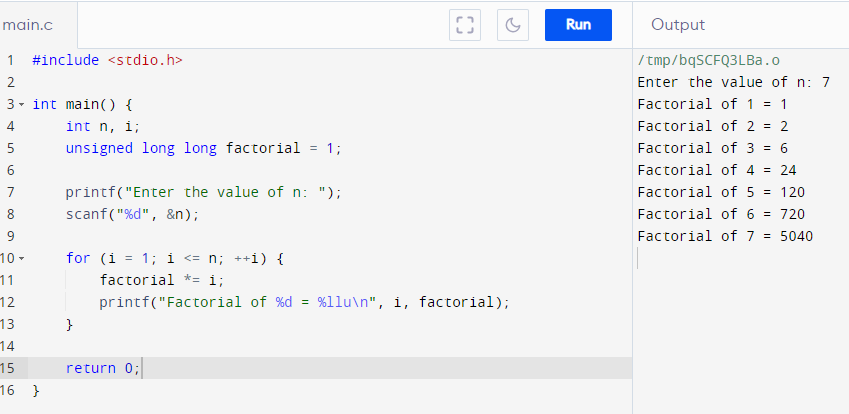
- Multiply factorial by i and store the result in factorial.

5. Print the factorial value for each i.

Dry run

i fact display

1. 1 1
2. 2 2
3. 6 6
4. 24 24



23

AIM: to generate the sum 1/fact(1) + 2/fact(2) + 3/fact(3) ... n/fact(n)

ALGORITHM:

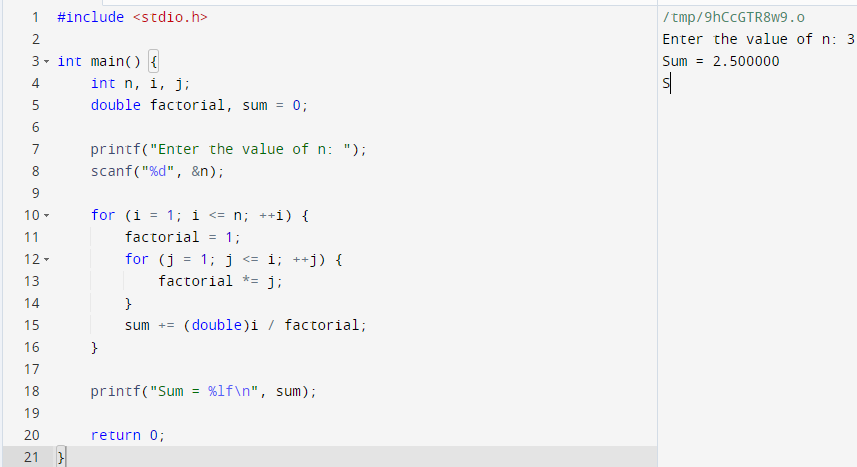
1. Declare variables: n, i, factorial, and sum.

2. Take input for n.

3. Initialize sum as 0.

4. Use a for loop to iterate from i = 1 to i <= n. - Initialize factorial as 1. - Use another for loop to calculate the factorial of i. - Multiply factorial by each number from 1 to i. - Add i/factorial to sum.

5. Print the value of sum.



24

AIM: program to display star patterns:

ALGORITHM:

1. Start

2. Declare the main function.

3. Declare variables to store the number of rows (`rows`), and loop counters (`i` and `j`).

4. Prompt the user to enter the number of rows.

5. Read and store the input value in the `rows` variable.

6. Implement the first pattern: the right-angled triangle.

- Use a nested loop structure.

- The outer loop iterates from 1 to `rows`.

- The inner loop prints `i` number of stars in each row.

- Print a new line after each inner loop iteration.

7. Implement the second pattern: the inverted right-angled triangle.

- Use a nested loop structure.

- The outer loop iterates from `rows` to 1.

- The inner loop prints `i` number of stars in each row.

- Print a new line after each inner loop iteration.

8. Implement the third pattern: the pyramid.

- Use a nested loop structure.

- The outer loop iterates from 1 to `rows`.

- The inner loop prints spaces (`rows - i`) followed by stars (`2 \* i - 1`) in each row.

- Print a new line after each inner loop iteration. 9. End the main function. 10. Compile and run the program.

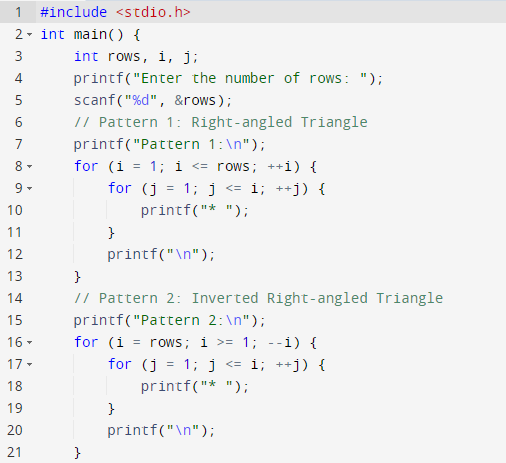
DRYRUN pattern3

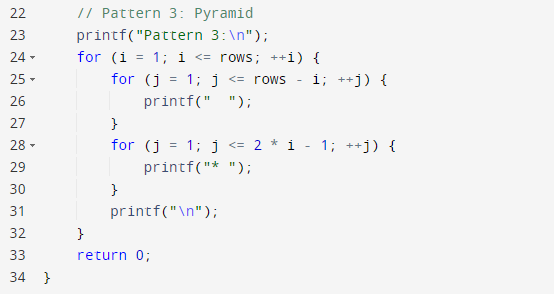
|  |  |  |  |
| --- | --- | --- | --- |
| i | j | k |  |
| 0 | 0 |  | No space |
| 0 |  | 3 | \* |
| 0 |  | 2 | \* |
| 0 |  | 1 | \* |
| 0 |  | 0 | Exit k |
| 1 | 0 |  | “ “ |
| 1 | 1 |  | Exit j |
| 1 |  | 3 | \* |
| 1 |  | 2 | \* |
| 1 |  | 1 | Exit k |
| 2 | 0 |  | “ “ |
| 2 | 1 |  | “ “ |
| 2 | 2 |  | Exit j |
| 2 |  | 3 | \* |
| 2 |  | 2 | Exit k |

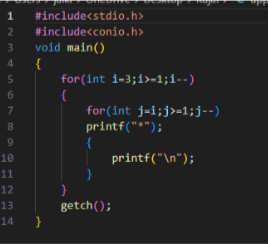
patternn 1

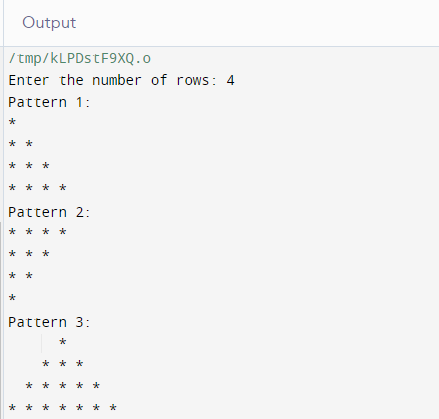
|  |  |  |  |
| --- | --- | --- | --- |
| i | j | k |  |
| 1 | 3 |  | “ “ |
| 1 | 2 |  | “ “ |
| 1 | 1 |  | Exit j |
| 1 |  | 0 | “\* “ |
| 1 |  | 1 | Exit k |
| 2 | 3 |  | “ “ |
| 2 | 2 |  | Exit j |
| 2 |  | 0 | “\* “ |
| 2 |  | 1 | “\* “ |
| 2 |  | 2 | Exit k |
| 3 | 3 |  | Exit j |
| 3 |  | 0 | “\* “ |
| 3 |  | 1 | “\* “ |
| 3 |  | 2 | “\* “ |
| 3 |  | 3 | Exit k |
| 3 | 3 |  | Exit i |

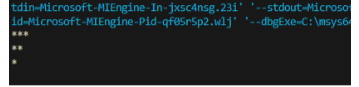
|  |  |  |  |
| --- | --- | --- | --- |
| i | j | k |  |
| 1 | 3 |  | “ “ |
| 1 | 2 |  | “ “ |
| 1 | 1 |  | Exit j |
| 1 |  | 0 | “\* “ |
| 1 |  | 1 | Exit k |
| 2 | 3 |  | “ “ |
| 2 | 2 |  | Exit j |
| 2 |  | 0 | “\* “ |
| 2 |  | 1 | “\* “ |
| 2 |  | 2 | Exit k |
| 3 | 3 |  | Exit j |
| 3 |  | 0 | “\* “ |
| 3 |  | 1 | “\* “ |
| 3 |  | 2 | “\* “ |
| 3 |  | 3 | Exit k |
| 3 | 3 |  | Exit i |





****





25.

AIM:ternary operators

DESCRIPTION:We use the ternary operator in C to run one code when the condition is true and another code when the condition is false.

SYNTAX:testCondition ? expression1 : expression 2;

The testCondition is a boolean expression that results in either true or false. If the condition is

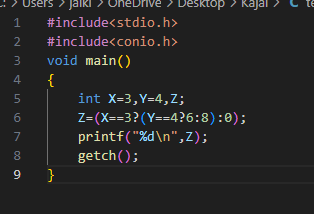
* true - expression1 (before the colon) is executed
* false - expression2 (after the colon) is executed

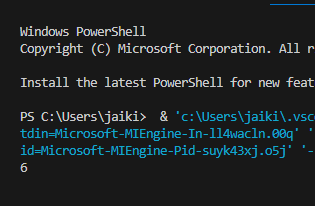
The ternary operator takes 3 operands (condition, expression1 and expression2). Hence, the name ternary operator.

ALGORITHM

1. Start
2. Declare X=3,Y=4,Z as integer
3. Z=(X==3?(Y==4?6:8):0)
4. Display the value
5. stop

coding





26

AIM:switch statement

