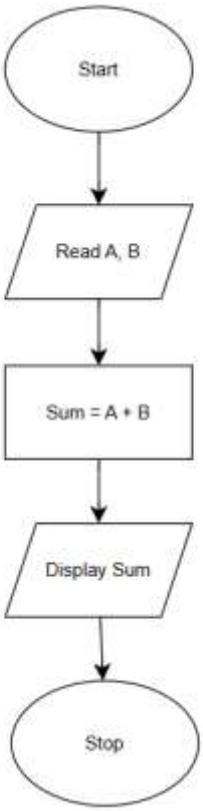
1. Draw a flow chart for adding two numbers.



Program

```
#include <stdio.h>
int main() {
    int A, B, Sum;
    printf("Enter two numbers: ");
    scanf("%d %d", &A,&B);
    Sum = A + B;
    printf("Sum = %d\n", Sum);
    return 0;
}
```

Algorithm

Start

Declare three integer variables: A, B, and Sum.

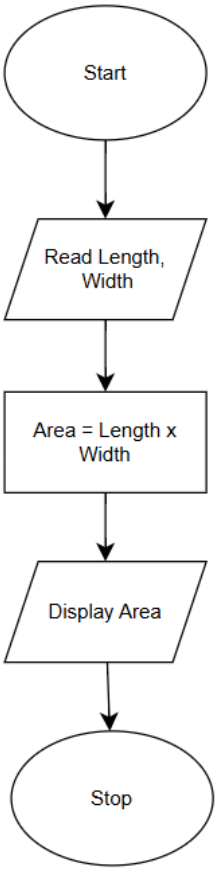
Read two integer values from the user and store them in variables A and B.

Calculate the sum of A and B, and store the result in Sum.

Display the sum

End

2. Draw a flow chart to print area of rectangle.



Program

```
#include <stdio.h>
int main() {
    float length, width, area;
    printf("Enter length and width of rectangle: ");
    scanf("%f  %f", &length, &width);
    area = length * width;
    printf("Area = %.2f\n", area);
    return 0;
```

}

Algorithm

Start

Declare three float variables: length, width, and area.

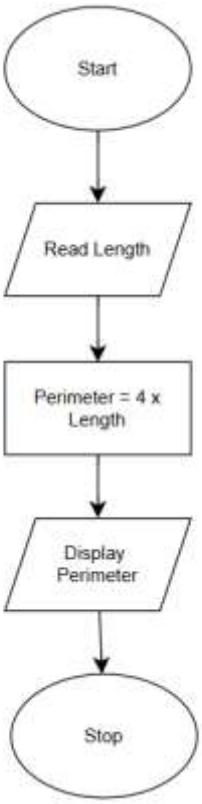
Read two float values from the user and store them in length and width.

Calculate the area using the formula: area = length × width.

Display the area

End

3. Draw a flow chart to print perimeter of a square.



Program

```
#include <stdio.h>
int main() {
    float Length, perimeter;
    printf("Enter the side length of the square: ");
```

```
    scanf("%f", &Length);
    perimeter = 4 * Length;
    printf("Perimeter = %.2f\n", perimeter);
    return 0;
}
```

Algorithm

Start

Declare two float variables: Length and perimeter.

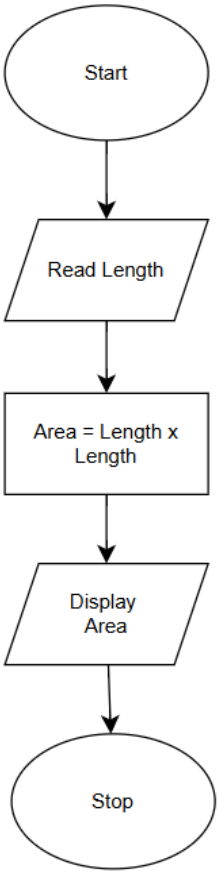
Read the float value from the user and store it in Length.

Calculate the perimeter using the formula: perimeter = 4 × Length.

Display the perimeter

End

4. Draw a flow chart to print area of square.



Program

```
#include <stdio.h>
int main() {
    float Length, area;
    printf("Enter the side length of the square: ");
    scanf("%f", &Length);
    area = Length * Length;
    printf("Area = %.2f\n", area);
    return 0;
}
```

Algorithm

Start

Declare two float variables: Length and area.

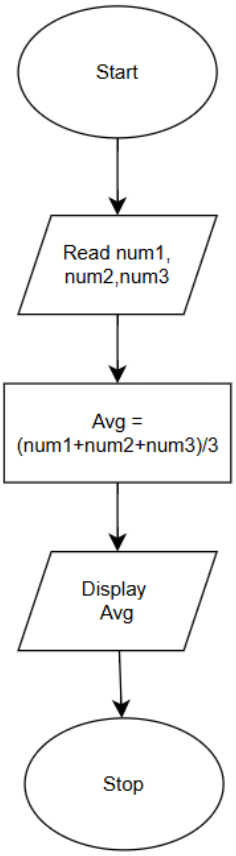
Read the float value from the user and store it in Length.

Calculate the area using the formula: area = Length × Length.

Display the area

End

5. Draw a flow chart to print the average of three numbers.



Program

```
#include <stdio.h>
int main() {
    float num1, num2, num3, avg;
    printf("Enter three numbers: ");
    scanf("%f %f %f", &num1, &num2, &num3);
    avg = (num1 + num2 + num3) / 3;
    printf("Average = %.2f\n", avg);
    return 0;
}
```

Algorithm

Start

Declare four float variables: num1, num2, num3, and avg.

Read three float values from the user and store them in num1, num2, and num3.

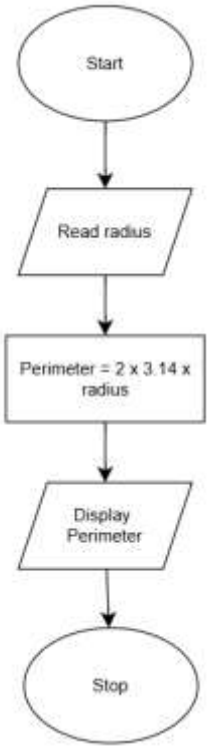
Calculate the average using the formula:

avg = (num1 + num2 + num3) / 3

Display the average

End

6. Draw a flow chart to print perimeter of a circle.



Program

```
#include <stdio.h>
int main() {
    float radius, perimeter;
    const float PI = 3.14;
    printf("Enter the radius of the circle: ");
    scanf("%f", &radius);
    perimeter = 2 * PI * radius;
    printf("Perimeter = %.2f\n", perimeter);
    return 0;
}
```

Algorithm

Start

Declare two float variables: radius and perimeter.

Define a constant PI = 3.14.

Read the float value from the user and store it in radius.

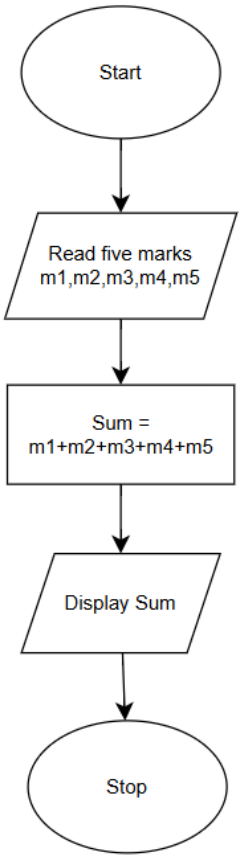
Calculate the perimeter using the formula:

perimeter = 2 × PI × radius

Display the perimeter

End

7. Draw a flow chart to print the sum of 5 subjects of students.



Program

```
#include <stdio.h>
int main() {
    float m1, m2, m3, m4, m5, Sum;
```

```
    printf("Enter marks of 5 subjects: ");
    scanf("%f  %f  %f  %f  %f", &m1, &m2, &m3, &m4, &m5);
    Sum = m1 + m2 + m3 + m4 + m5;
    printf("Total Sum = %.2f\n", Sum);
    return 0;
}
```

Algorithm

Start

Declare six float variables: m1, m2, m3, m4, m5, and sum.

Read five float values from the user and store them in m1 to m5.

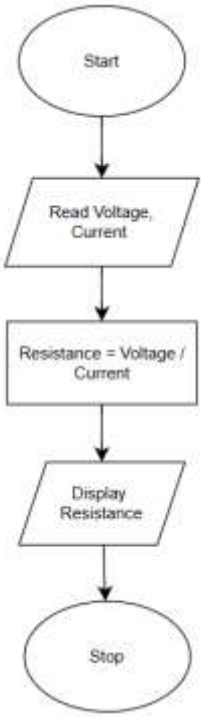
Calculate the total using the formula:

sum = m1 + m2 + m3 + m4 + m5

Display the sum

End

8. Draw a flow chart to print the value of resistance, when voltage and current values are provided.



Program

```
#include <stdio.h>
int main() {
   float voltage, current, resistance;
   printf("Enter Voltage (V): ");
   scanf("%f", &voltage);
   printf("Enter Current (I): ");
   scanf("%f", &current);
   if (current == 0) {
      printf("Error: Current cannot be zero!\n");
   }
   else {
       resistance = voltage / current;
       printf("Resistance (R) = %.2f Ohms\n", resistance);}
       return 0;
}
```

Algorithm

Start

Declare three float variables: voltage, current, and resistance.

Read float value into voltage.

Read float value into current.

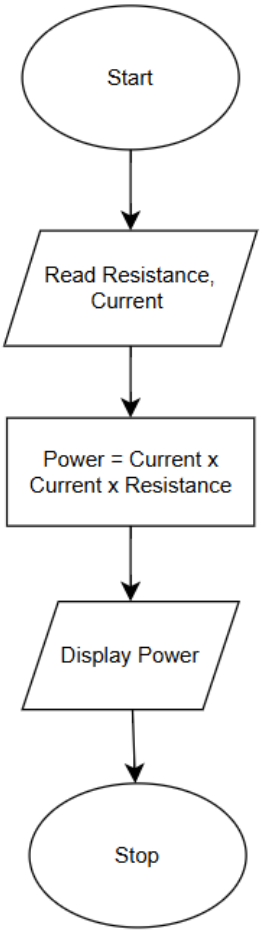
Check if current is zero:

- If yes, display "Error: Current cannot be zero!"
- Else, calculate resistance using:
  resistance = voltage / current

Display Resistance

End

9. Draw a flow chart to print the power when the current and resistance values are provided.



Program

```
#include <stdio.h>
int main() {
    float current, resistance, power;
    printf("Enter Current : ");
    scanf("%f", &current);
    printf("Enter Resistance : ");
    scanf("%f", &resistance);
    power = current * current * resistance;
    printf("Power (P) = %.2f Watts\n", power);
    return 0;
}
```

Algorithm

Start

Declare three float variables: current, resistance, and power.

Read float value into current.

Read float value into resistance.

Calculate power using the formula:

power = current × current × resistance

Display power

End

10. Draw a flow chart to take the values of i1, i2, i3, i4, i5, i6 current values following from the point and calculate sum of current.

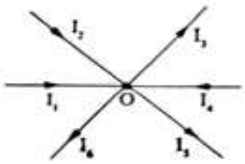
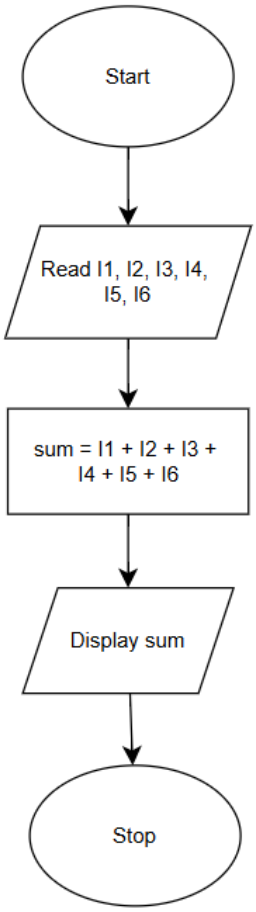


Fig. 4.1



Program

```
#include <stdio.h>
int main() {
    float I1, I2, I3, I4, I5, I6, sum;
    printf("Enter values of I1 to I6 (in Amps):\n");
    scanf("%f %f %f %f %f %f", &I1, &I2, &I3, &I4, &I5, &I6);
    sum = I1 + I2 + I3 + I4 + I5 + I6;
    printf("Sum of currents = %.2f A\n", sum);
    return 0;
}
```

Algorithm

Start

Declare seven float variables: I1, I2, I3, I4, I5, I6, and sum.

Read six float values from the user and store them in I1 to I6.

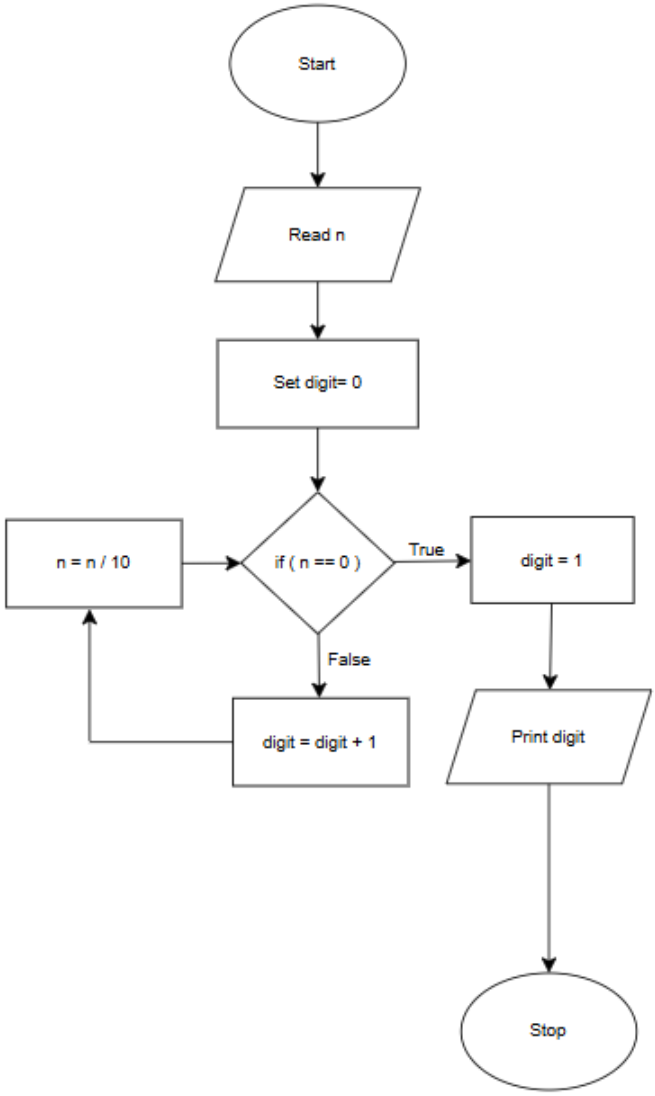
Calculate the sum using:

sum = I1 + I2 + I3 + I4 + I5 + I6

Display sum

End

11. Draw a flow chart to read the number of digits in given number.



Program

```
#include <stdio.h>
int main() {
    int n, digit = 0;
    printf("Enter a number: ");
    scanf("%d", &n);
    if (n == 0) {
        digit = 1;
    } else {
        while (n != 0) {
```

```
            digit++;
            n = n / 10;
        }
    }
    printf("Number of digits: %d\n", digit);
    return 0;
}
```

Algorithm

Start

Declare two integer variables: n and digit, initialize digit = 0.

Read the integer input into n.

Check if n == 0:

- If true, set digit = 1.

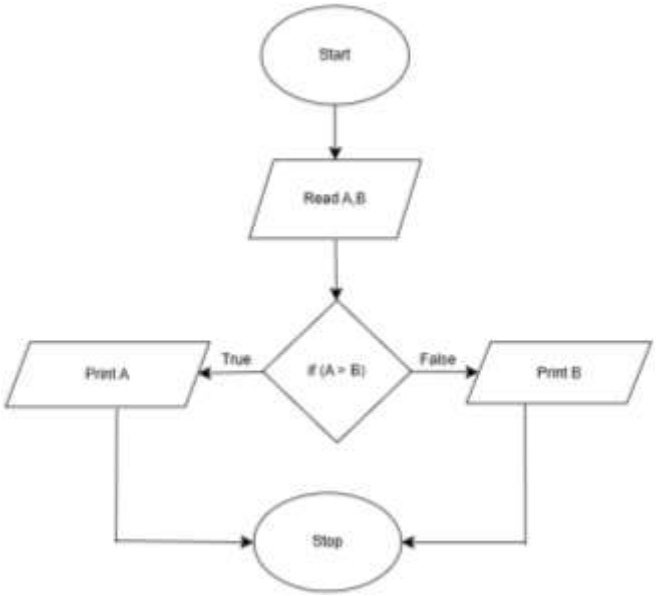
Else, repeat until n becomes 0:

- Increment digit by 1.
- Divide n by 10 (drop the last digit).

Display digit

End

12. Draw a flow chart to read two numbers from user and print bigger value.



Program

```
#include <stdio.h>
int main() {
   int A, B;
   printf("Enter two numbers: ");
   scanf("%d  %d", &A, &B);
   if (A > B) {
      printf("Bigger value is: %d\n", A);
   } else {
      printf("Bigger value is: %d\n", B);
   }
   return 0;
}
```

Algorithm

Start

Declare two integer variables: A and B.

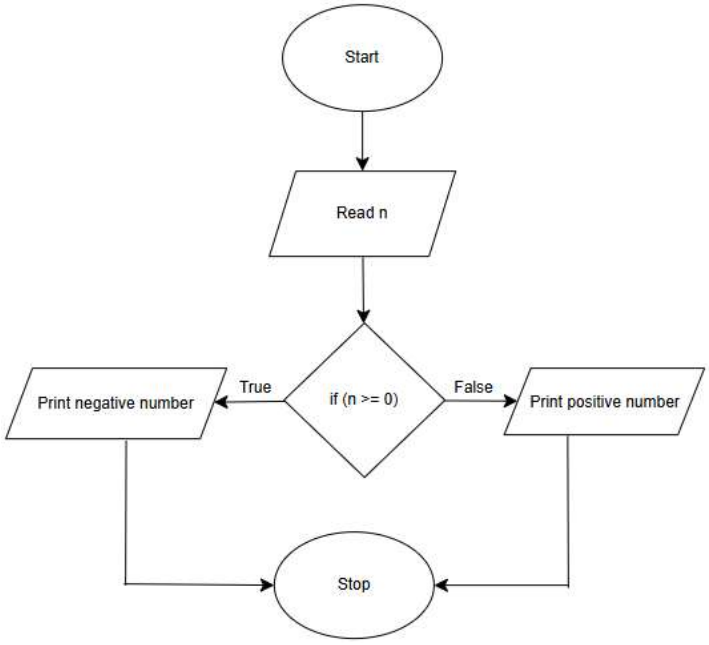
Read two integer values from the user and store them in A and B.

Compare A and B:

- If A > B, then:
  Display: "Bigger value is: A"
- Else:
  Display: "Bigger value is: B"

End

13. Draw a flow chart to print whether the given number is positive or negative.



Program

```
#include <stdio.h>
int main() {
    int n;
    printf("Enter a number: ");
    scanf("%d", &n);
    if (n >= 0)
        printf("The number is Positive.\n");
    else
        printf("The number is Negative.\n");

    return 0;
}
```

Algorithm

Start

Declare an integer variable n.

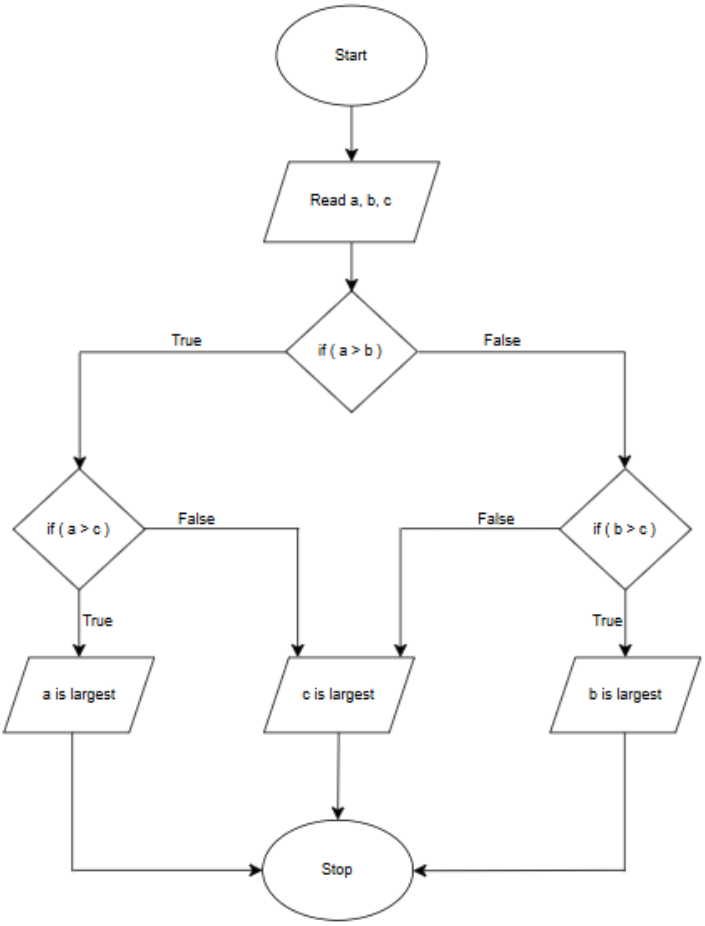
Read the integer input into n.

Check if n >= 0:

- If true, display: "The number is Positive."
- Else, display: "The number is Negative."

End

14. Draw a flow chart to read the three numbers from the user and print the biggest value.



Program

```
#include <stdio.h>

int main() {

    int A, B, C;
```

```
    printf("Enter three numbers: ");
    scanf("%d %d %d", &A, &B, &C);
    if (A > B) {
        if (A > C)
            printf("Biggest = %d\n", A);
        else
            printf("Biggest = %d\n", C);
    } else {
        if (B > C)
            printf("Biggest = %d\n", B);
        else
            printf("Biggest = %d\n", C);
    }
    return 0;
}
```

Algorithm

Start

Declare three integer variables: A, B, and C.

Read the three integers into A, B, and C.

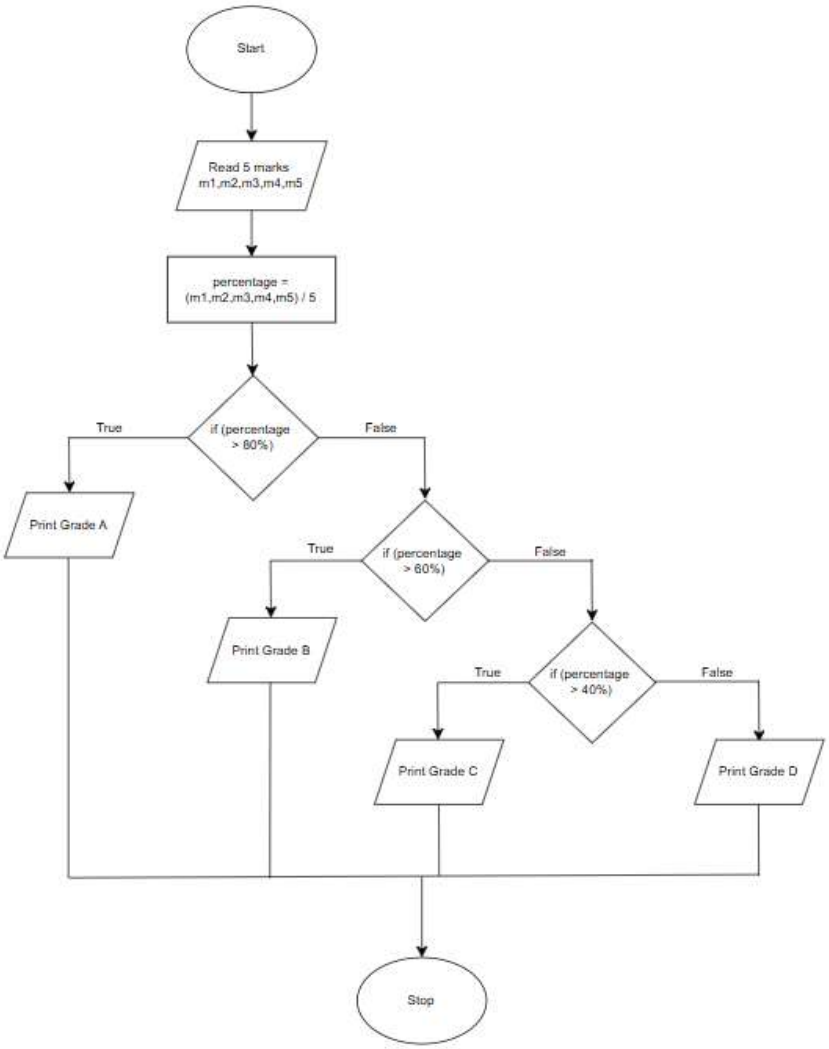
Check if A > B:

- If true, check if A > C:
  If true, A is the biggest.
  Else, C is the biggest.
- Else, check if B > C:
  If true, B is the biggest.
  Else, C is the biggest.

Display the biggest number.

End

15. Draw a flow chart to get marks of 5 subjects of a student and print grade A is above 80%, B is above 60%, C is above 40%, D is below 40%.



Program

```
#include <stdio.h>
int main() {
    float m1, m2, m3, m4, m5, percentage;
    printf("Enter marks of 5 subjects:\n");
    scanf("%f %f %f %f %f", &m1, &m2, &m3, &m4, &m5);
    percentage = (m1 + m2 + m3 + m4 + m5) / 5;
    printf("Percentage: %.2f%%\n", percentage);
    if (percentage > 80)
        printf("Grade: A\n");
```

```
    else if (percentage > 60)
        printf("Grade: B\n");
    else if (percentage > 40)
        printf("Grade: C\n");
    else
        printf("Grade: D\n");
    return 0;
}
```

Algorithm

Start

Declare six float variables: m1, m2, m3, m4, m5, and percentage.

Read five float values into m1, m2, m3, m4, and m5.

Calculate the percentage:

percentage = (m1 + m2 + m3 + m4 + m5) / 5

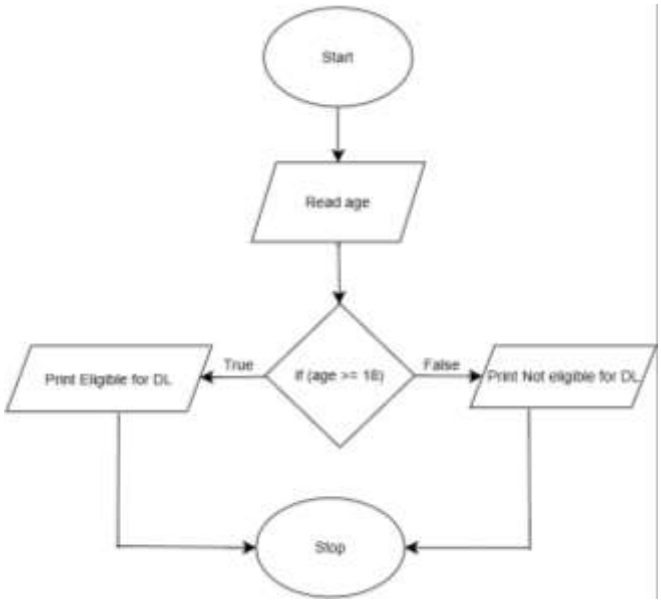
Display the percentage:

Determine grade using conditions:

- If percentage > 80, display "Grade: A"
- Else if percentage > 60, display "Grade: B"
- Else if percentage > 40, display "Grade: C"
- Else, display "Grade: D"

End

16. Draw a flow chart to get age of a person to print whether he is eligible for DL or not.



Program

```
#include <stdio.h>
int main() {
    int age;
    printf("Enter your age: ");
    scanf("%d", &age);
   if (age >= 18)
        printf("You are eligible for Driving License.\n");
    else
        printf("You are not eligible for Driving License.\n");
    return 0;
}
```

Algorithm

Start

Declare an integer variable age.

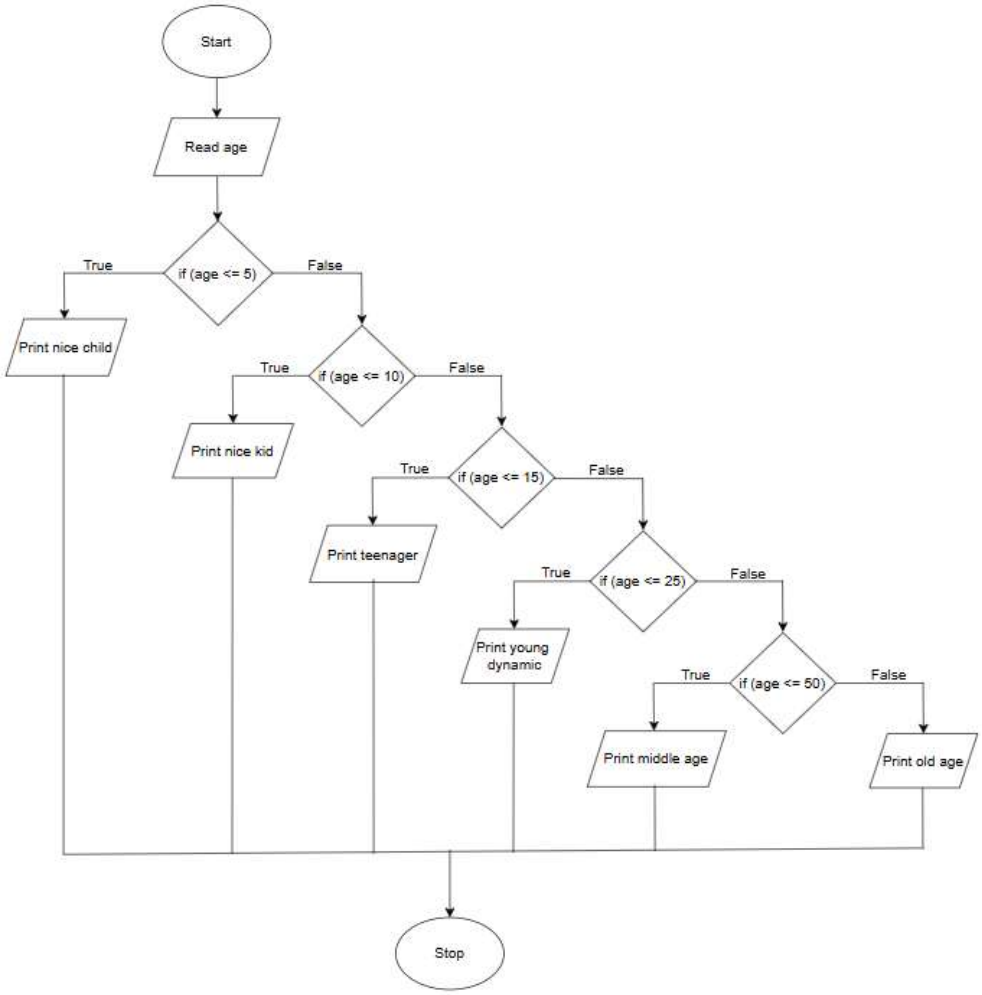
Read the value into age.

Check if age >= 18:

- If true, display: "You are eligible for Driving License."
- Else, display: "You are not eligible for Driving License."

End

17. Draw a flow chart to get age of a person to print greeting message if age<=5 nice child, age<=10 nice kid, age<=15 teenagers, age<=25 young dynamic, age<=50 middle age, age<=50 old.



Program

```
#include <stdio.h>
int main() {
    int age;
    printf("Enter age: ");
    scanf("%d", &age);
    if (age <= 5)
        printf("Nice child\n");
```

```
    else if (age <= 10)
        printf("Nice kid\n");
    else if (age <= 15)
        printf("Teenager\n");
    else if (age <= 25)
        printf("Young & dynamic\n");
    else if (age <= 50)
        printf("Middle-aged\n");
    else
        printf("Old\n");
    return 0;
}
```

Algorithm

Start

Declare an integer variable age.

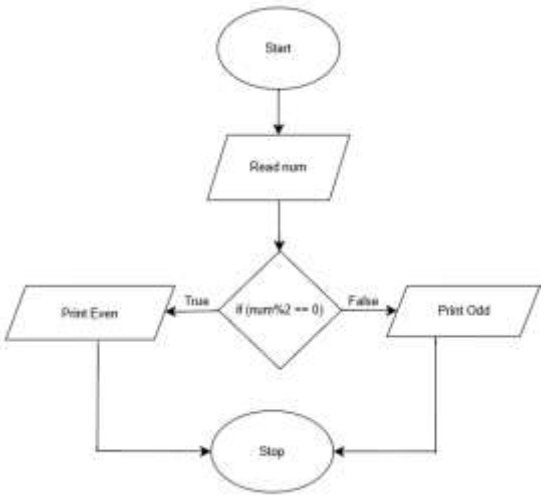
Read the value into age.

Check conditions in sequence:

- If age <= 5, display: "Nice child"
- Else if age <= 10, display: "Nice kid"
- Else if age <= 15, display: "Teenager"
- Else if age <= 25, display: "Young & dynamic"
- Else if age <= 50, display: "Middle-aged"
- Else, display: "Old"

End

18. Draw a flow chart to get whether the given number is even or odd.



Program

```
#include <stdio.h>
int main() {
    int n;
    printf("Enter a number: ");
    scanf("%d", &n);
    if (n % 2 == 0)
        printf("Even\n");
    else
        printf("Odd\n");
    return 0;
}
```

Algorithm

Start

Declare an integer variable n.

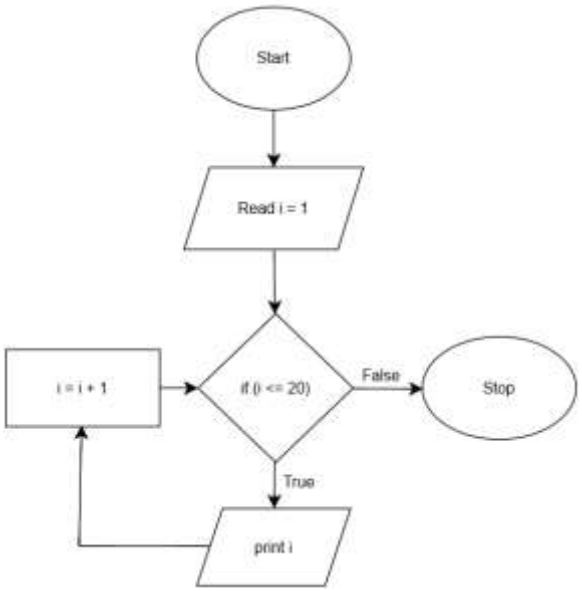
Read the value into n.

Check if n % 2 == 0:

- If true, display: "Even"
- Else, display: "Odd"

End

19. Draw a flow chart to print first 20 natural numbers.



Program

```
#include <stdio.h>
int main() {
   int i = 1;
   while (i <= 20) {
      printf("%d\n", i);
      i++;
   }
   return 0;
}
```

Algorithm

Start

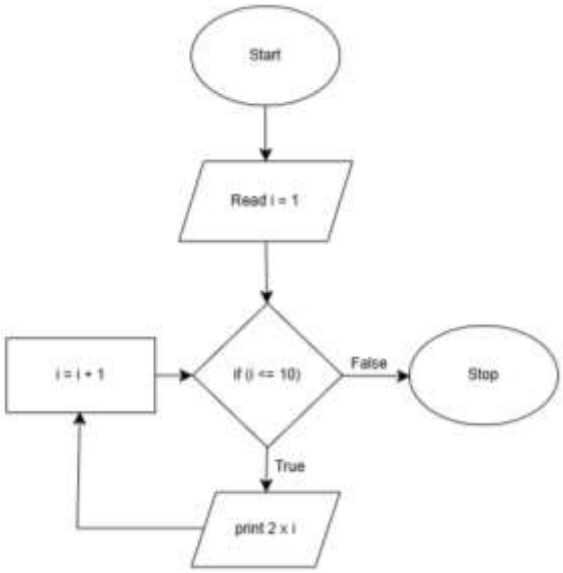
Initialize an integer variable i = 1

Repeat the following steps while i <= 20:

- Display the value of i
- Increment i by 1

End

20. Draw a flow chart to print first 10 even numbers



Program

```
#include <stdio.h>
int main() {
   int i = 1;
   while (i <= 10) {
      printf("%d\n", 2 * i);
      i++;
   }
   return 0;
}
```

Algorithm

Start

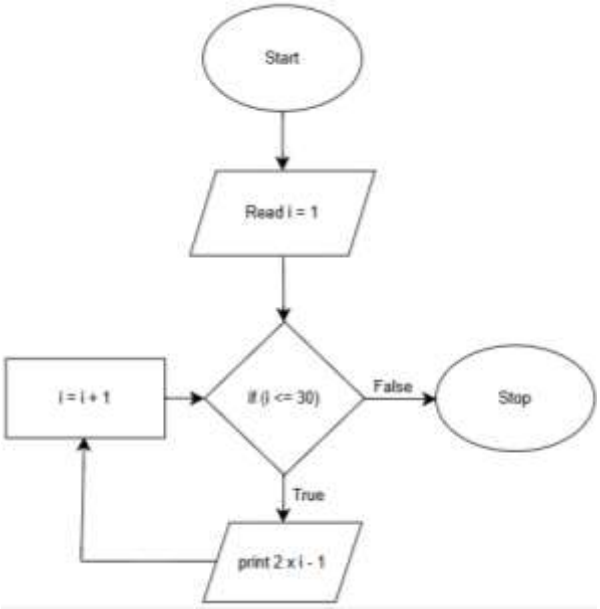
Initialize an integer variable i = 1

Repeat the following steps while i <= 10:

- Calculate 2 * i
- Display the result
- Increment i by 1

End

21. Draw a flow chart to print first 30 odd numbers.



Program

```
#include <stdio.h>
int main() {
   int i = 1;
   while (i <= 30) {
      printf("%d\n", 2 * i - 1);
      i++;
   }
   return 0;
}
```

Algorithm

Start

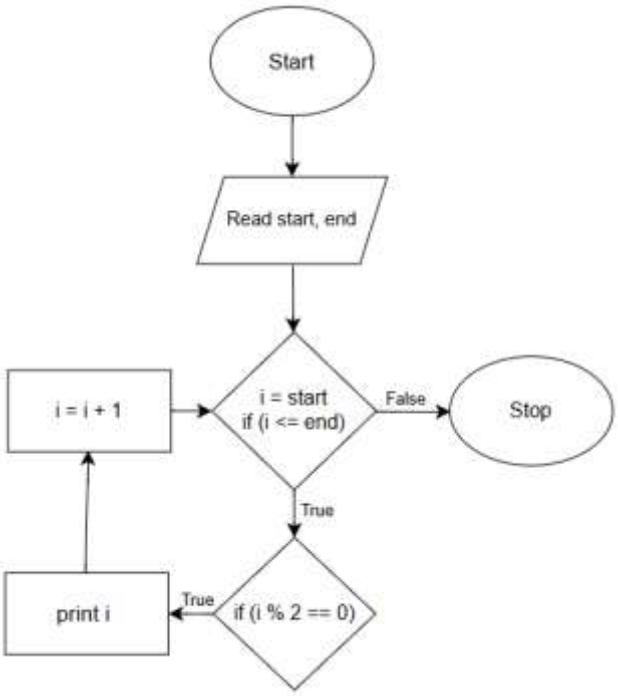
Initialize integer variable i = 1

Repeat the following steps while i <= 30:

- Calculate 2 * i - 1
- Display the result
- Increment i by 1

End

22. Draw a flow chart to print all the even numbers in range.



Program

```
#include <stdio.h>
int main() {
    int start, end;
    printf("Enter the start and end of range: ");
    scanf("%d %d", &start, &end);
    for (int i = start; i <= end; i++) {
        if (i % 2 == 0) {
            printf("%d ", i);
        }
    }
    return 0;
}
```

Algorithm

Start

Declare two integer variables: start and end

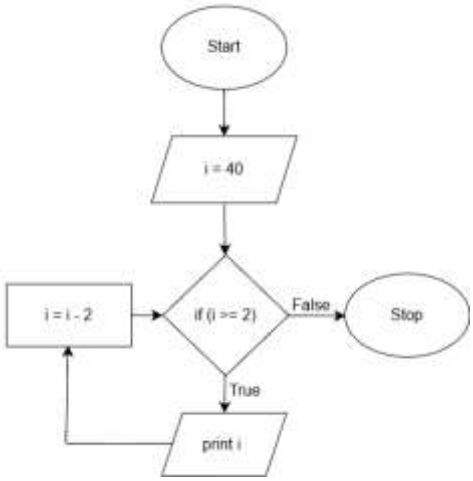
Read values into start and end

Loop from i = start to i <= end:

- If i % 2 == 0 (i is even):
  Display i

End

23. Draw a flow chart to print first 20 even numbers in reverse order.



Program

```
#include <stdio.h>
int main() {
   for (int i=40;i>=2;i--){
      if(i%2==0){
      printf("%d\n", i);
      }
   }
   return 0;
}
```

Algorithm

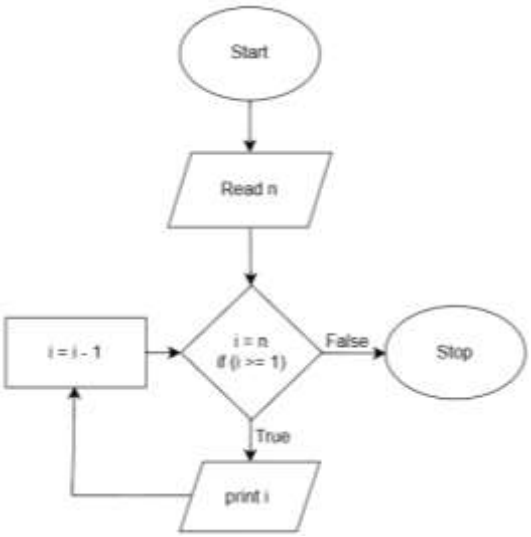
Start

Loop from i = 40 down to i >= 2, decreasing i by 1 each time:

- If i % 2 == 0 (i is even):
  Display i

End

24. Draw a flow chart to print first n natural numbers in reverse order.



Program

```
#include <stdio.h>
int main() {
   int n;
   printf("Enter the value of n: ");
   scanf("%d", &n);
   for (int i=n;i>=1;i--)
   {
      printf("%d\n", i);
   }
   return 0;
}
```

Algorithm

Start

Declare an integer variable n

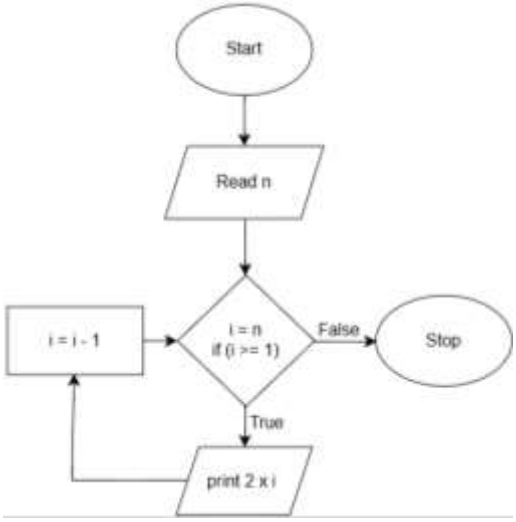
Read value into n

Loop from i = n down to i >= 1:

- Display i

End

25. Draw a flow chart to print first n even numbers in reverse order.



Program

```
#include <stdio.h>
int main() {
   int n;
   printf("Enter the value of n: ");
   scanf("%d", &n);
   for (int i=n;i>=1;i--)
   {
      printf("%d\n", 2 * i);
   }
   return 0;
}
```

Algorithm

Start

Declare an integer variable n

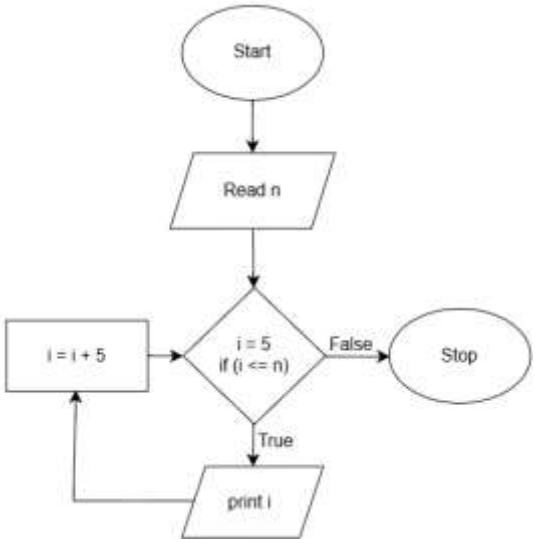
Read value into n

Loop from i = n down to i >= 1:

- Compute 2 * i
- Display the result

End

26. Draw a flow chart to print the series of 5,10,15,20,.....n numbers.



Program

```
#include <stdio.h>
int main() {
   int n;
   printf("Enter the value of n: ");
   scanf("%d", &n);
   for (int i=5;i<=n;i+=5)
   {
      printf("%d\n", i);
   }
   return 0;
}
```

Algorithm

Start

Declare an integer variable n

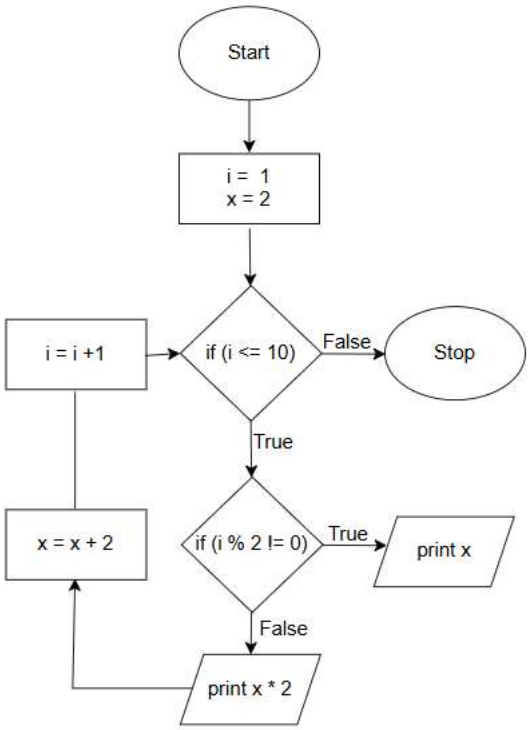
Read value into n

Loop from i = 5 to i <= n, incrementing by 5:

- Display i

End

27. Draw a flow chart to print the series of numbers 2,4,4,8,6,12,8,16,10,20.



Program

```
#include <stdio.h>
int main() {
   int i, x = 2;
   for (i = 1; i <= 10; i++) {
      if (i % 2 != 0) {
         printf("%d ", x);
      } else {
         printf("%d ", x * 2);
         x += 2;
      }
   }
   return 0;
}
```

Algorithm

Start

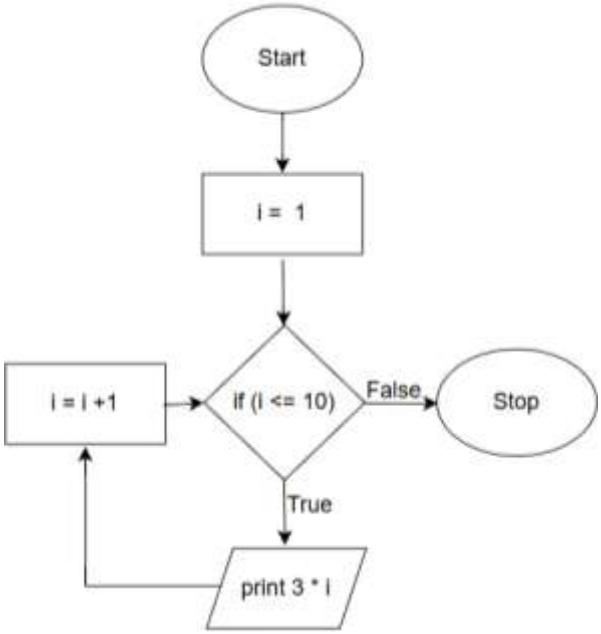
Initialize x = 2

Loop i from 1 to 10:

- If i is odd, print x
- If i is even, print 2 * x and increment x by 2

End

28. Draw a flow chart to print multiplication table of 3.



Program

```
#include <stdio.h>
int main() {
   int i;
   for (i = 1; i <= 10; i++) {
      printf("3 x %d = %d\n", i, 3 * i);
   }
   return 0;
}
```

Algorithm

Start

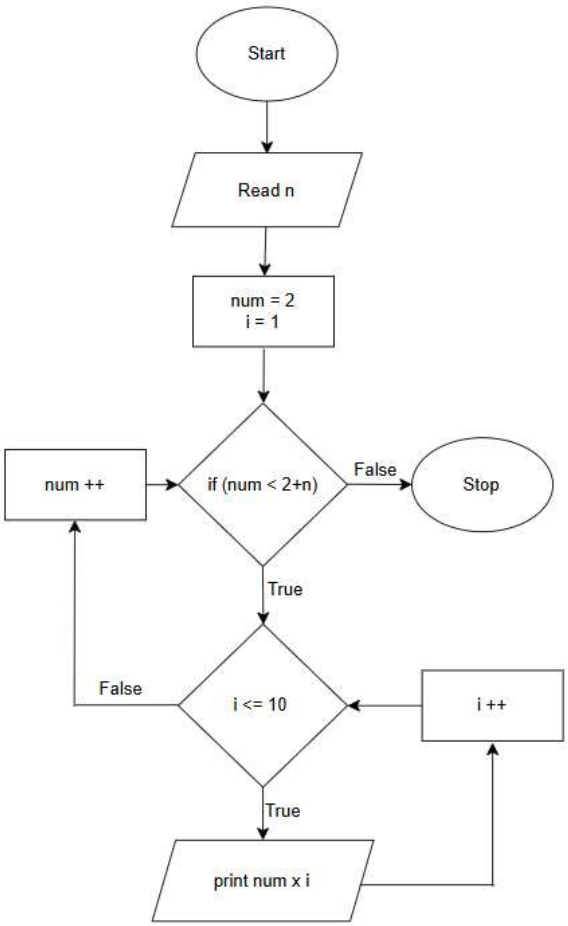
Declare integer variable i

Loop from i = 1 to i <= 10:

- Calculate 3 * i
- Print the result in format: "3 x i = result"

End

29. Draw a flow chart to print first n multiplication table from 2.



Program

```
#include <stdio.h>
int main() {
    int n;
    printf("Enter how many tables to print from 2: ");
    scanf("%d", &n);
    for (int num = 2; num < 2 + n; num++) {
        printf("\nTable of %d:\n", num);
        for (int i = 1; i <= 10; i++) {
            printf("%d x %d = %d\n", num, i, num * i);
```

```
        }
    }
    return 0;
}
```

Algorithm

Start

Declare an integer variable n

Read n from user

Loop num from 2 to 2 + n - 1:

- Display: "Table of num"
- Nested loop i from 1 to 10:

    Compute num × i

    Display in format: "num x i = result"

End

30. Draw a flow chart to print given pattern

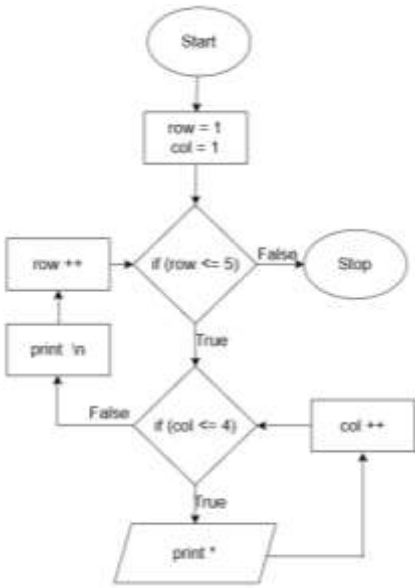
*** * * ***

*** * * ***

*** * * ***

*** * * ***

*** * * ***



Program

```
#include <stdio.h>
int main() {
    for (int row = 1; row <= 5; row++) {
        for (int col = 1; col <= 4; col++) {
            printf("* ");
        }
        printf("\n");
    }
    return 0;
}
```

Algorithm

Start

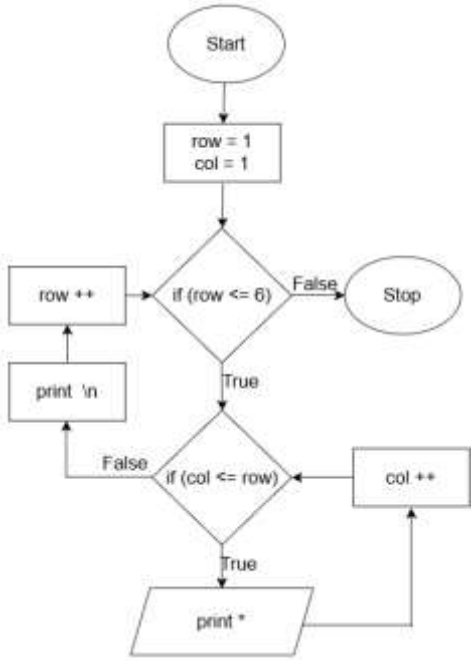
Loop row from 1 to 5:

- Inside that, loop col from 1 to 4:
  Print *
- After inner loop, print a newline (\n)

End

31. Draw a flow chart to print given pattern

```
*
* *
* * *
* * * *
* * * * *
* * * * * *
```



Program

```
#include <stdio.h>
int main() {
    for (int row = 1; row <= 6; row++) {
        for (int col = 1; col <= row; col++) {
            printf("* ");
        }
        printf("\n");
    }
    return 0;
}
```

Algorithm

Start

Loop row from 1 to 6:

- Inside that, loop col from 1 to row:
  Print * (with space)
- After inner loop, print a newline (\n)

End

32. Draw a flow chart to print given pattern

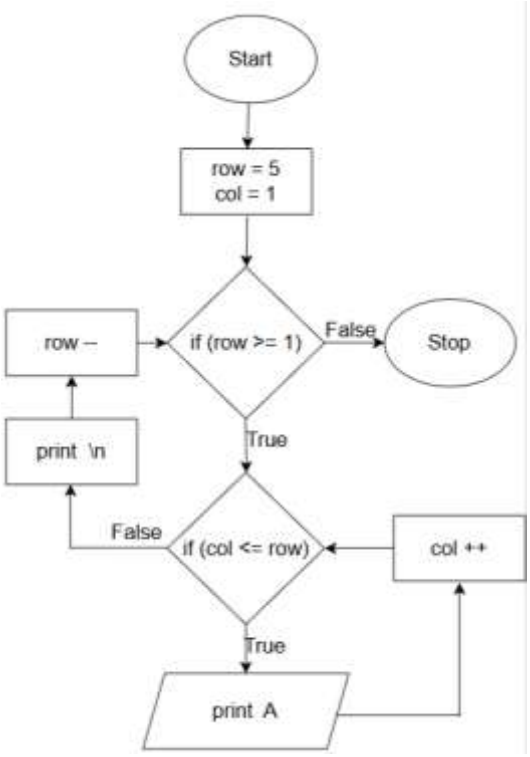
AAAAA

AAAA

AAA

AA

A



Program

```
#include <stdio.h>
int main() {
    for (int row = 5; row >= 1; row--) {
        for (int col = 1; col <= row; col++) {
            printf("A");
        }
        printf("\n");
    }
    return 0;
}
```

Algorithm

Start

Loop row from 5 down to 1:

- Inside that, loop col from 1 to row:
  Print 'A'
- After inner loop, print newline (\n)

End

33. Draw a flow chart to print given pattern

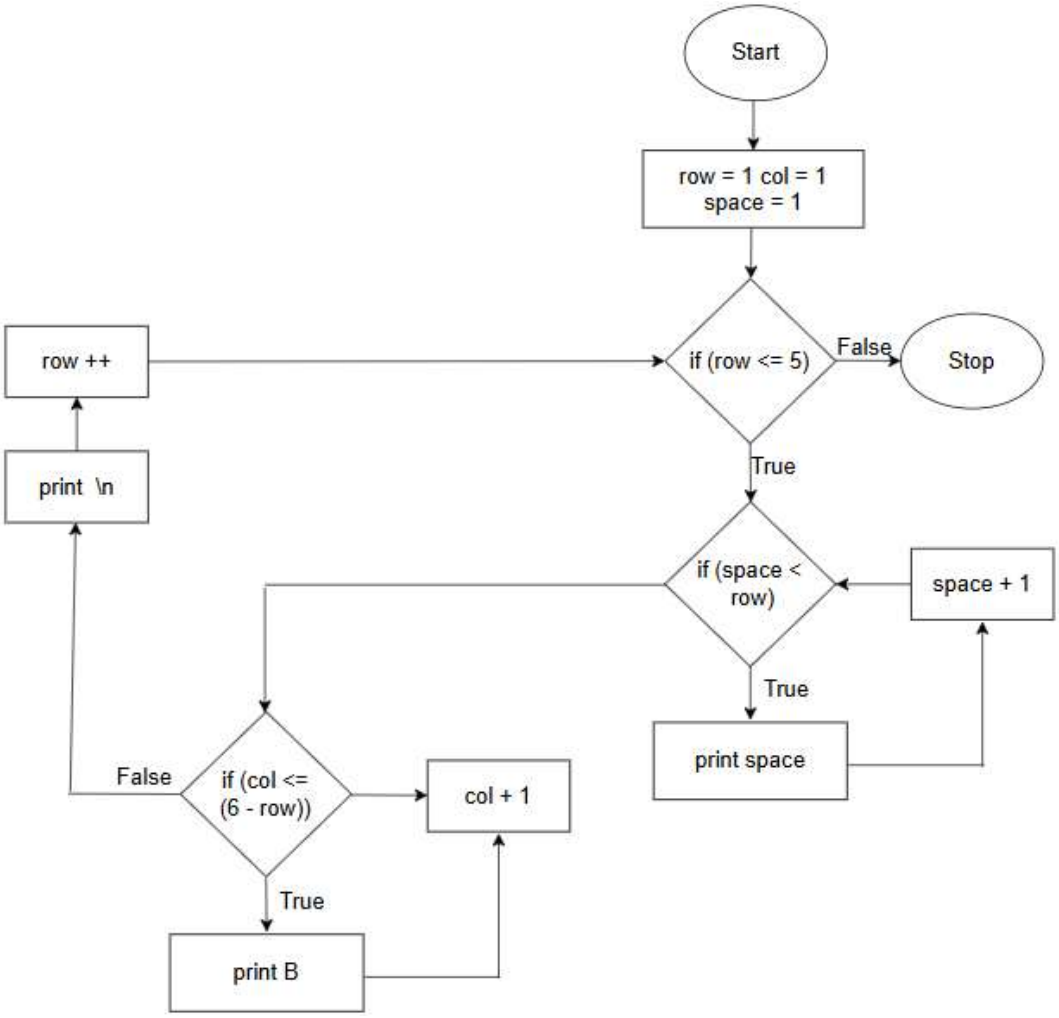
BBBBB

BBBB

BBB

BB

B



Program

```
#include <stdio.h>
int main() {
    for (int row = 1; row <= 5; row++) {
        for (int space = 1; space < row; space++) {
            printf(" ");
        }
        for (int col = 1; col <= 6 - row; col++) {
            printf("B");
        }
        printf("\n");
    }
    return 0;
}
```

Algorithm

Start

Loop row from 1 to 5:

- Print row - 1 spaces using a loop
- Print 6 - row B characters using another loop
- Print newline

End

34. Draw a flow chart to print given pattern

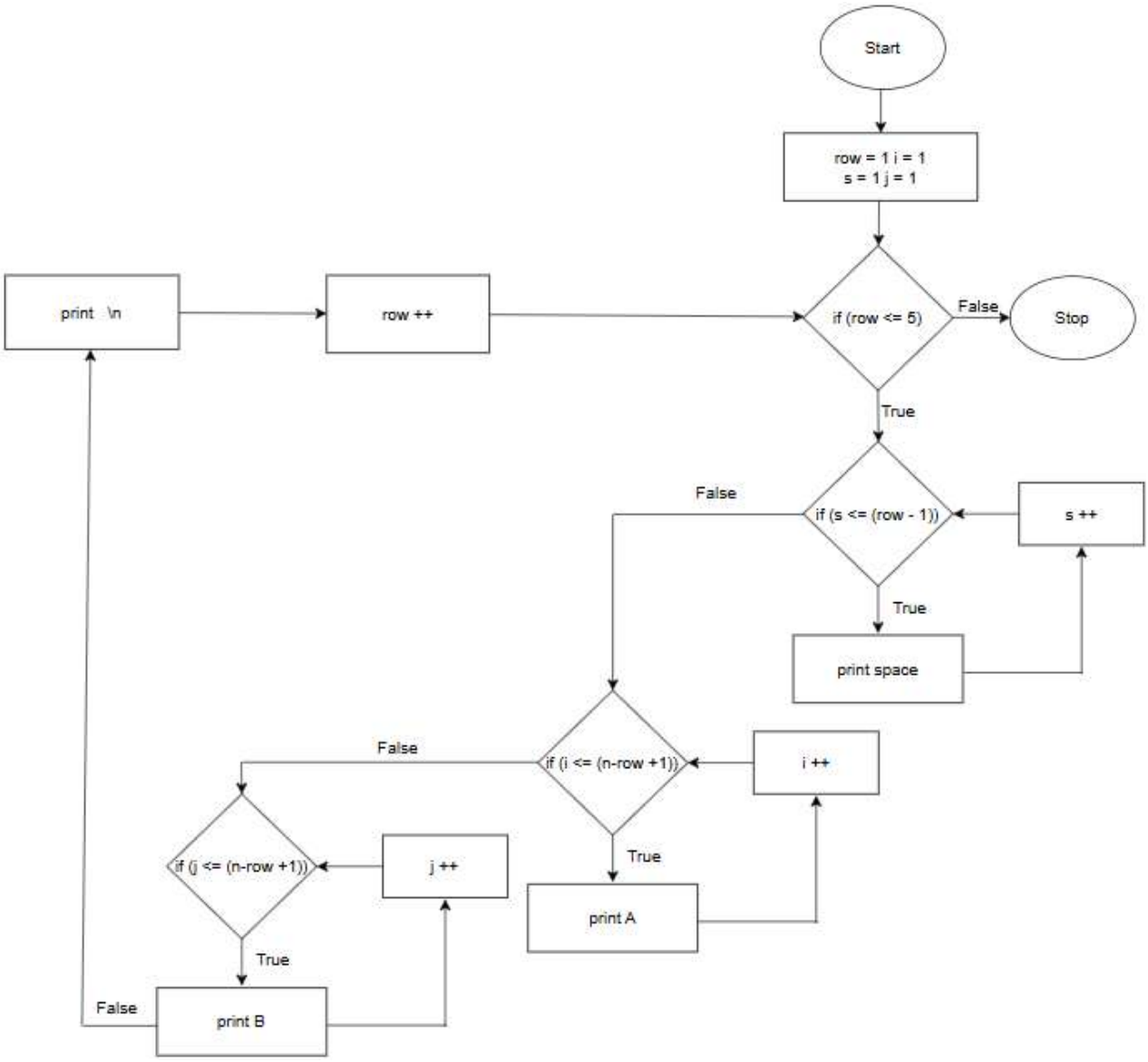
AAAAABBBBB

AAAABBBB

AAABBB

AABB

AB



Program

```
#include <stdio.h>
int main() {
    int n = 5;
    for (int row = 1; row <= n; row++) {
        for (int s = 1; s <= row - 1; s++) {
            printf(" ");
        }
        for (int i = 1; i <= n - row + 1; i++) {
            printf("A");
```

```
        }
        for (int j = 1; j <= n - row + 1; j++) {
            printf("B");
        }
        printf("\n");
    }
    return 0;
}
```

Algorithm

Start

Set n = 5

Repeat for row from 1 to n:

- Print row - 1 spaces:
  Loop from s = 1 to row - 1
  Print " "
- Print n - row + 1 'A' characters:
  Loop from i = 1 to n - row + 1
  Print "A"
- Print n - row + 1 'B' characters:
  Loop from j = 1 to n - row + 1
  Print "B"
- Print newline (\n)

End

35. Draw a flow chart to print given pattern

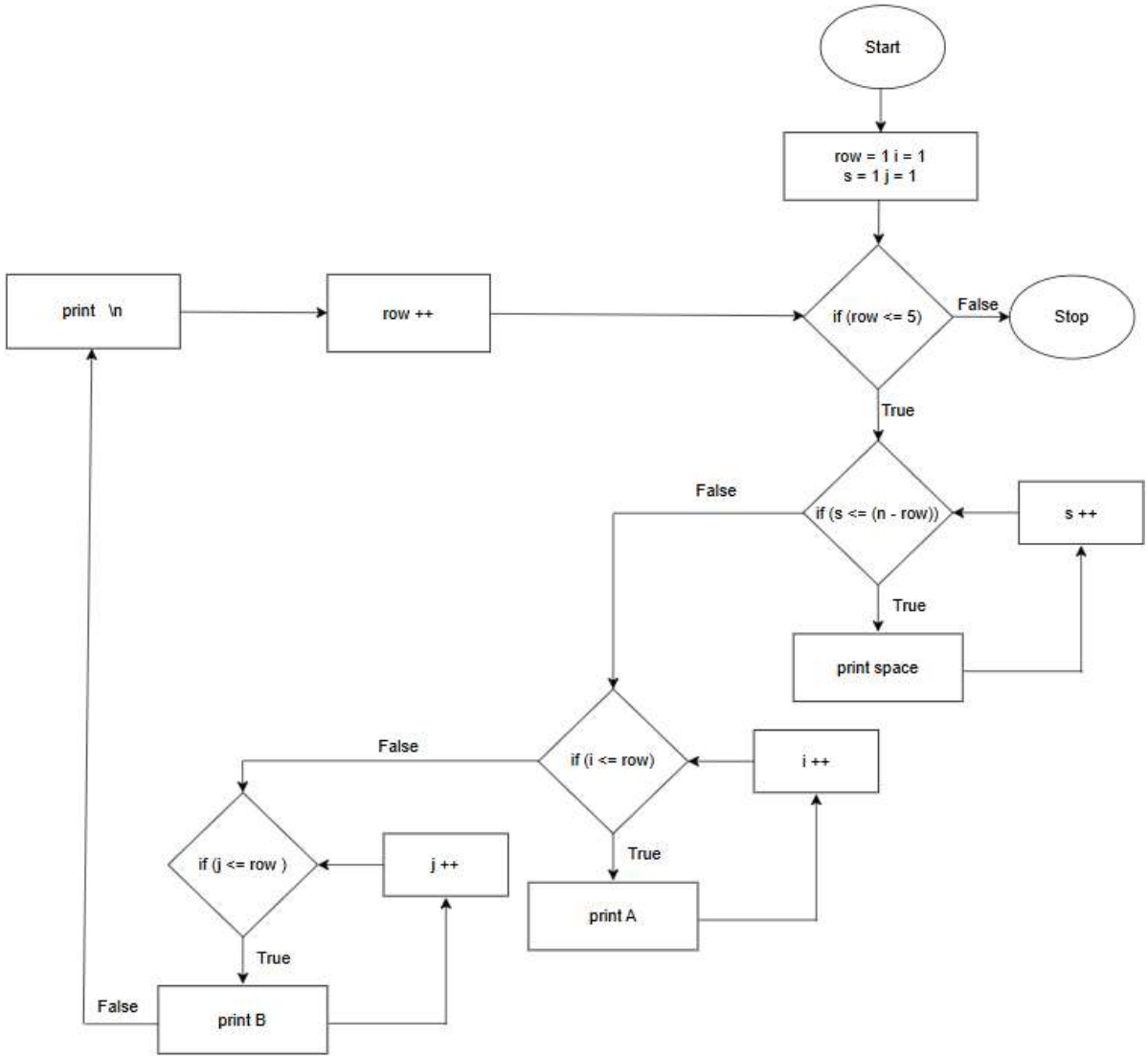
AB

AABB

AAABBB

AAAABBBB

AAAAABBBBB



Program

```
#include <stdio.h>
int main() {
    int n = 5;
```

```
    for (int row = 1; row <= n; row++) {
        for (int s = 1; s <= n - row; s++) {
            printf(" ");
        }
        for (int i = 1; i <= row; i++) {
            printf("A");
        }
        for (int j = 1; j <= row; j++) {
            printf("B");
        }
        printf("\n");
    }
    return 0;
}
```

Algorithm

Start

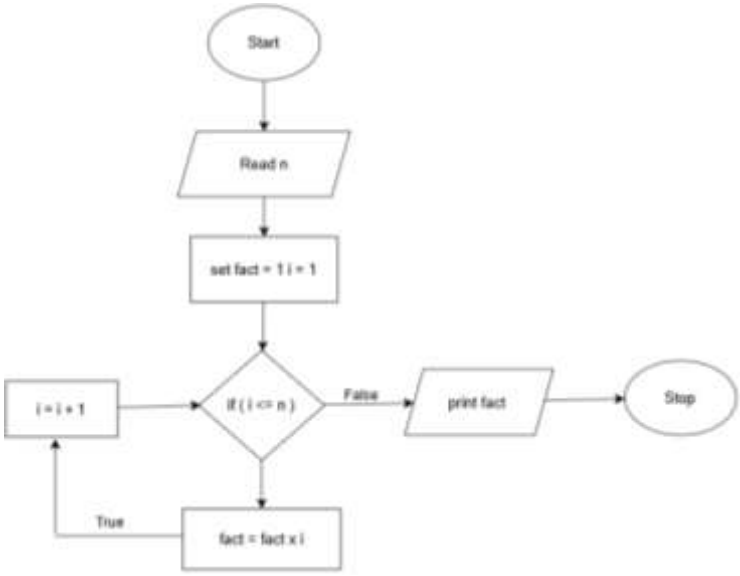
Set n = 5 (or take it as user input)

Repeat for row = 1 to n:

- Print n - row spaces:
  Loop s = 1 to n - row → print " "
- Print row times 'A':
  Loop i = 1 to row → print "A"
- Print row times 'B':
  Loop j = 1 to row → print "B"
- Print newline

End

36. Draw a flow chart to print factorial of given number



Program

```
#include <stdio.h>
int main() {
    int n, i;
    int fact = 1;

    printf("Enter a number: ");
    scanf("%d", &n);
    for (i = 1; i <= n; i++) {
        fact *= i;
    }
    printf("Factorial of %d is %d\n", n, fact);
    return 0;
}
```

Algorithm

Start

Declare: Integer n, I, fact = 1

Read value into n

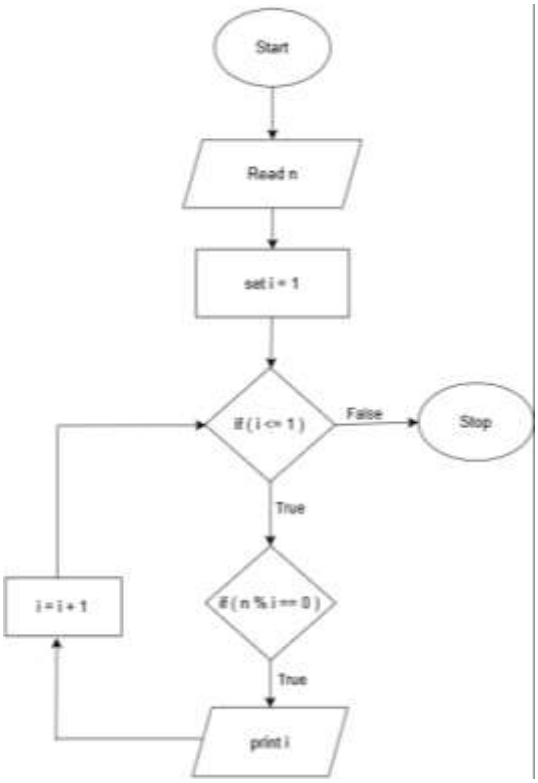
Repeat for i = 1 to n:

- Multiply fact by i → fact = fact * i

After loop, print: "Factorial of n is fact"

End

37. Draw a flow chart to print factors of given numbers



Program

```
#include <stdio.h>
int main() {
    int n;
    printf("Enter a number: ");
    scanf("%d", &n);
    printf("Factors of %d are: ", n);
    for (int i = 1; i <= n; i++) {
        if (n % i == 0) {
            printf("%d ", i);
        }
```

```
    }
    printf("\n");
    return 0;
}
```

Algorithm

Start

Declare an integer variable n

Read input into n

Display message: "Factors of n are:"

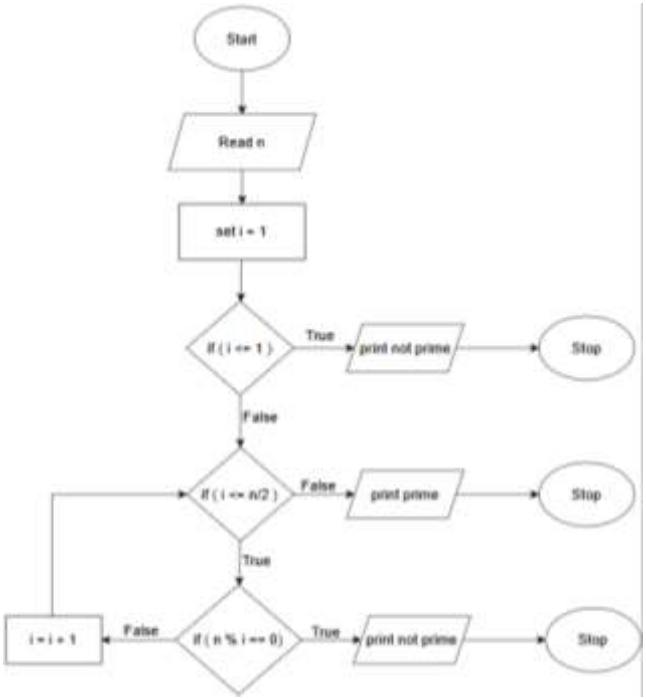
Loop from i = 1 to n:

- If n % i == 0, then:
  i is a factor → print i

Print newline

End

38. Draw a flow chart to print whether the given number is prime or not.



Program

```
#include <stdio.h>
int main() {
```

```
    int n, i, isPrime = 1;
    printf("Enter a number: ");
    scanf("%d", &n);
    if (n <= 1) {
        printf("Not Prime\n");
        return 0;
    }
    for (i = 2; i <= n / 2; i++) {
        if (n % i == 0) {
            isPrime = 0;
            break;
        }
    }
    if (isPrime)
        printf("Prime\n");
    else
        printf("Not Prime\n");
    return 0;
}
```

Algorithm

Start

Declare: Integer n, isPrime = 1

Read value into n

If n <= 1, print "Not Prime" and exit

Loop from i = 2 to n / 2:

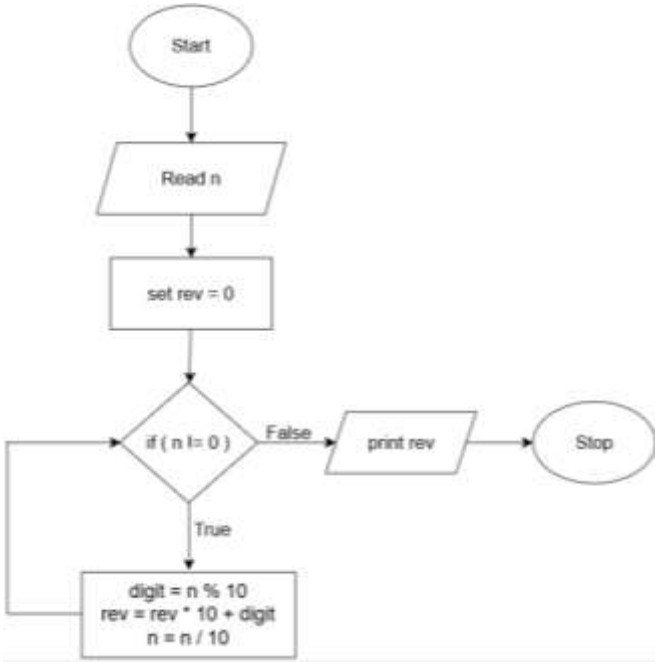
- If n % i == 0:
  Set isPrime = 0
  Break the loop

After the loop:

- If isPrime == 1, print "Prime"
- Else, print "Not Prime"

End

39. Draw a flow chart to print reverse of a given number.



Program

```
#include <stdio.h>
int main() {
   int n, digit, rev = 0;

   printf("Enter a number: ");
   scanf("%d", &n);
   while (n != 0) {
      digit = n % 10;
      rev = rev * 10 + digit;
      n = n / 10;
   }
   printf("Reversed number = %d\n", rev);
```

```
    return 0;
}
```

Algorithm

Start

Declare: n, digit, rev = 0

Read value into n

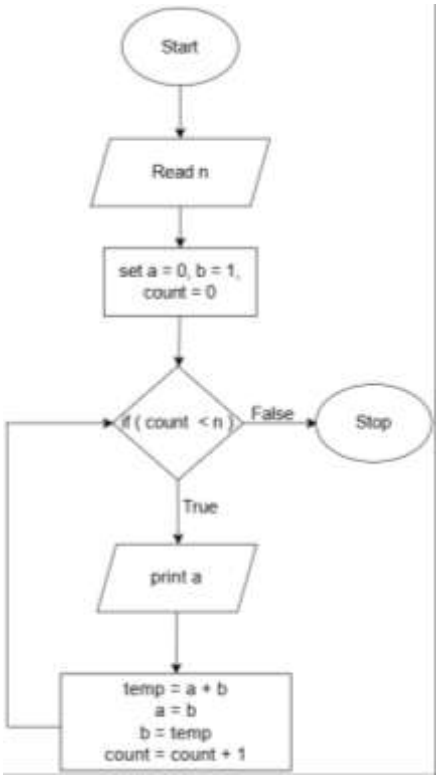
Loop while n != 0:

- digit = n % 10
- rev = rev * 10 + digit
- n = n / 10 → remove last digit

Print: "Reversed number = rev"

End

40. Draw a flow chart to print Fibonacci series.



Program

```
#include <stdio.h>
int main() {
    int n, a = 0, b = 1, temp, count = 0;
```

```
    printf("Enter number of terms: ");
    scanf("%d", &n);
    while (count < n) {
        printf("%d ", a);
        temp = a + b;
        a = b;
        b = temp;
        count++;
    }
    printf("\n");
    return 0;
}
```

Algorithm

Start

Declare: n, a = 0, b = 1, temp, count = 0

Read value into n

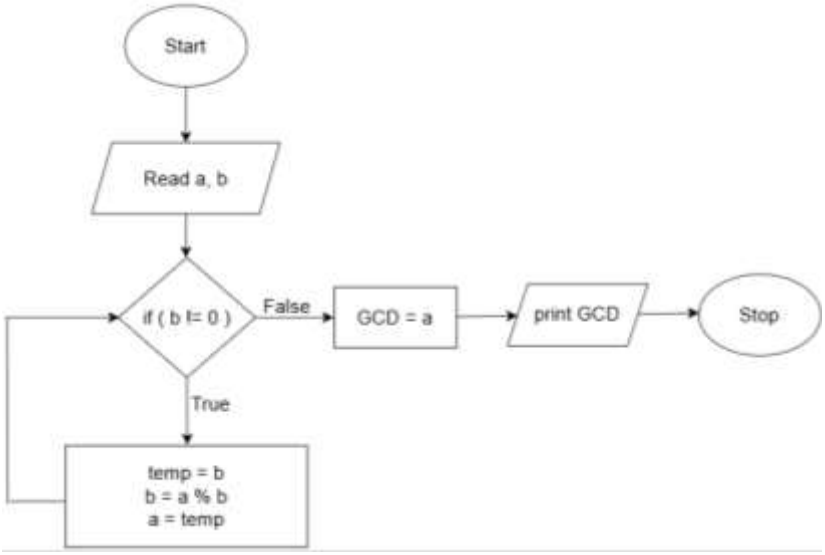
Loop while count < n:

- Print current term: a
- Calculate next term: temp = a + b
- Update values:
  a = b
  b = temp
- Increment count

Print newline

End

41. Draw a flow chart to find GCD of two given numbers



Program

```
#include <stdio.h>
int main() {
    int a, b, temp;
    printf("Enter two numbers: ");
    scanf("%d %d", &a, &b);

    while (b != 0) {
        temp = b;
        b = a % b;
        a = temp;
    }
    printf("GCD is %d\n", a);
    return 0;
}
```

Algorithm

Start

Declare three integers: a, b, temp

Input two integers a and b from the user

While b is not equal to 0:

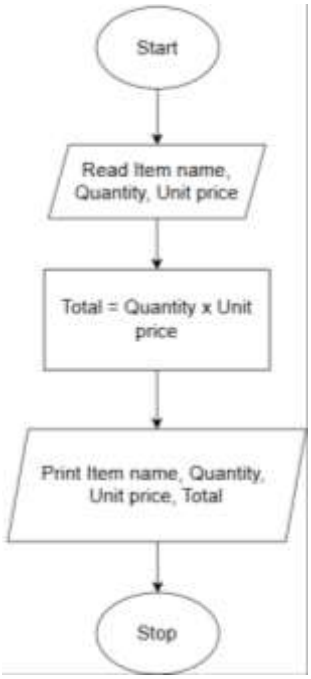
- temp ← b
- b ← a % b
- a ← temp

When loop ends, a contains the GCD

Print: "GCD is a"

End

42. Draw a flow chart to print bill of shopping items, quantity, unit price to be entered.



Program

```
#include <stdio.h>
int main() {
    char item[50];
    int quantity;
    float unit_price, total;
    printf("Enter item name: ");
    scanf("%s", item);
    printf("Enter quantity: ");
    scanf("%d", &quantity);
```

```
    printf("Enter unit price: ");
    scanf("%f", &unit_price);
    total = quantity * unit_price;
    printf("\n---- BILL ----\n");
    printf("Item      : %s\n", item);
    printf("Quantity  : %d\n", quantity);
    printf("Unit Price: %.2f\n", unit_price);
    printf("Total     : %.2f\n", total);
    return 0;
}
```

Algorithm

Start

Declare:item[50], quantity, unit_price, total

Input item name

Input quantity

Input unit price

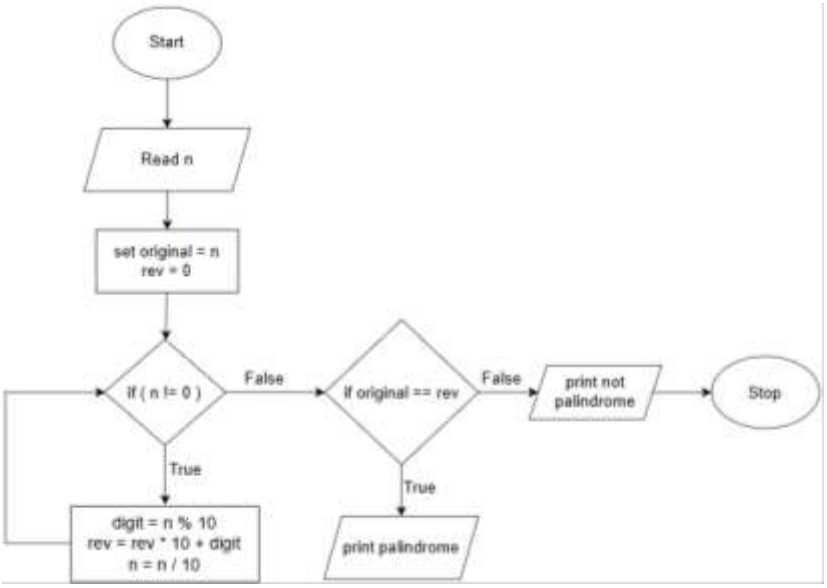
Calculate total: total = quantity × unit_price

Display the bill:

- Item name
- Quantity
- Unit price
- Total price

End

43. Draw a flow chart to print check whether the given number is palindrome or not.



Program

```
#include <stdio.h>
int main() {
    int n, original, rev = 0, digit;
    printf("Enter a number: ");
    scanf("%d", &n);
    original = n;
    while (n != 0) {
        digit = n % 10;
        rev = rev * 10 + digit;
        n = n / 10;
    }
    if (original == rev)
        printf("Palindrome\n");
    else
        printf("Not Palindrome\n");
    return 0;
}
```

Algorithm

Start

Input an integer n

Store the original number in a variable original

Initialize rev = 0

Repeat while n != 0:

- digit = n % 10 → extract last digit
- rev = rev * 10 + digit → build reversed number
- n = n / 10 → remove last digit

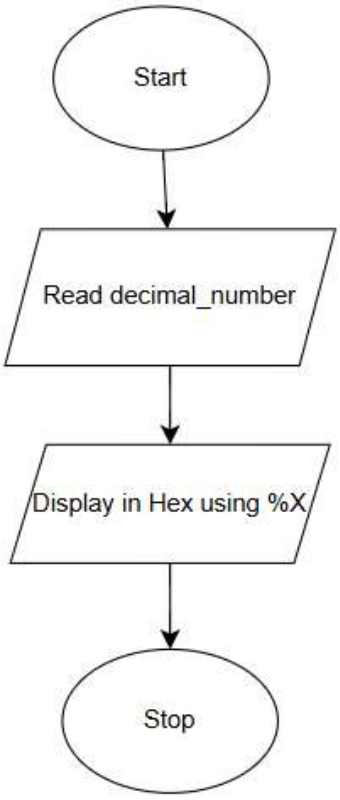
Compare:

- If original == rev, it's a Palindrome
- Else, it's Not Palindrome

Print result

End

44. Draw a flow chart to print Hexa value of given decimal no.



Program

```
#include <stdio.h>
int main() {
    int decimal;
    printf("Enter a decimal number: ");
    scanf("%d", &decimal);
    printf("Hexadecimal: %X\n", decimal);
    return 0;
}
```

Algorithm

Start

Declare an integer variable decimal

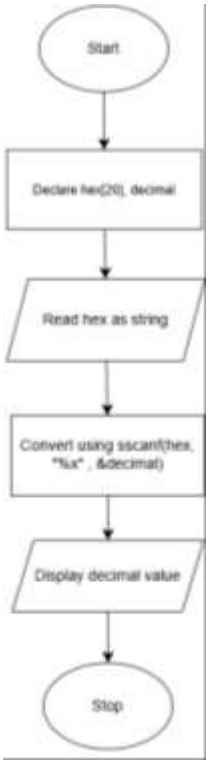
Read input into decimal

Convert the number to hexadecimal using %X format specifier

Print the hexadecimal value

End

45. Draw a flow chart to print decimal value of hexa number.



Program

```
#include <stdio.h>
int main() {
    int decimal;
    char hex[20];
    printf("Enter a hexadecimal number: ");
    scanf("%s", hex);
    sscanf(hex, "%x", &decimal);
    printf("Decimal value: %d\n", decimal);
    return 0;
}
```

Algorithm

Start

Declare a string hex[20] and integer decimal

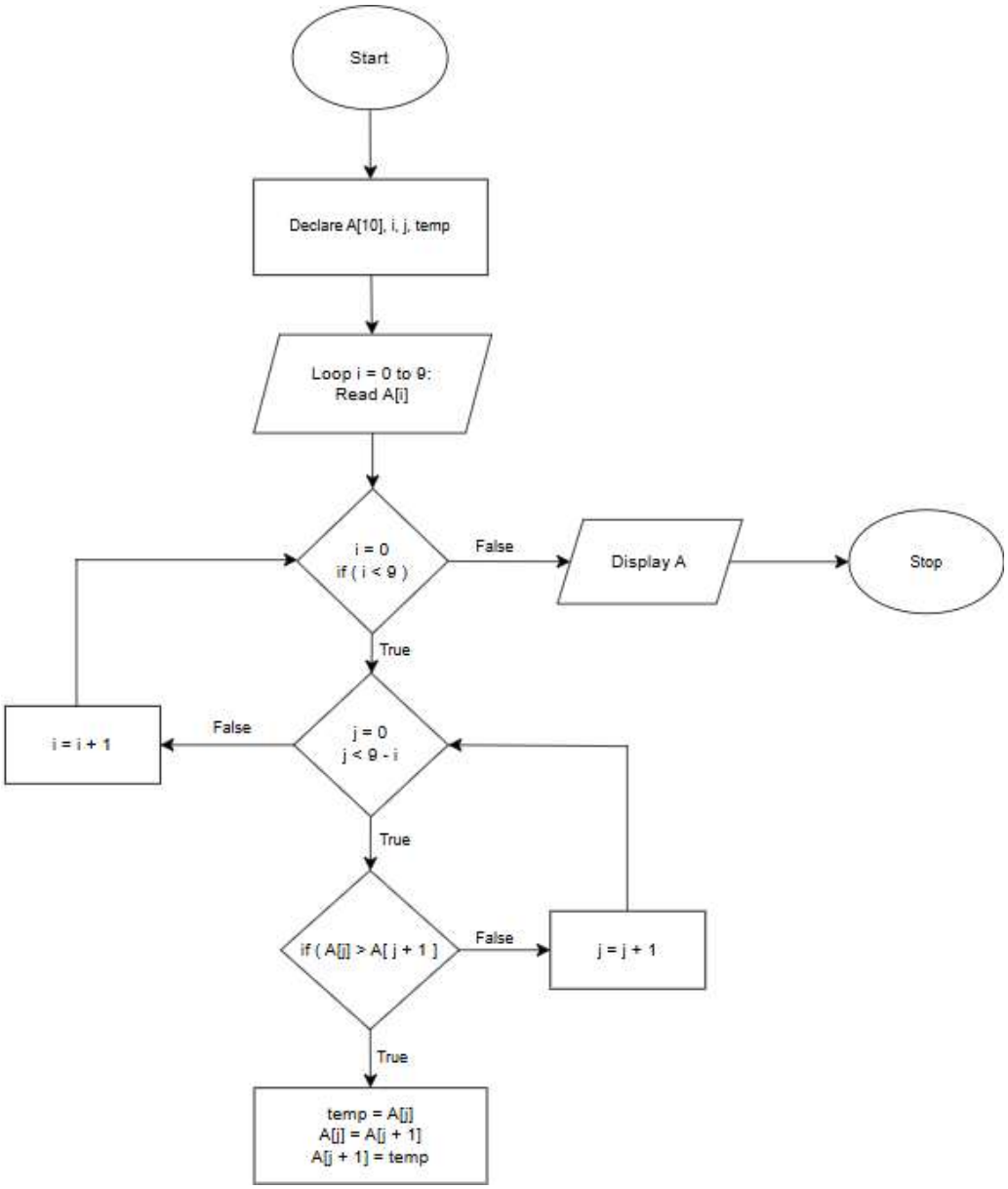
Input the hex number as a string

Convert hex to decimal using sscanf(hex, "%x", &decimal)

Print the decimal result

End

46. Draw a flow chart to get 10 numbers from the user and print the ascending order A[0]



Program

```
#include <stdio.h>
int main() {
    int A[10], i, j, temp;
    printf("Enter 10 numbers:\n");
    for (i = 0; i < 10; i++) {
        scanf("%d", &A[i]);
    }
```

```
    for (i = 0; i < 9; i++) {
        for (j = 0; j < 9 - i; j++) {
            if (A[j] > A[j + 1]) {
                temp = A[j];
                A[j] = A[j + 1];
                A[j + 1] = temp;
            }
        }
    }
    printf("Sorted in ascending order:\n");
    for (i = 0; i < 10; i++) {
        printf("A[%d] = %d\n", i, A[i]);
    }
    return 0;
}
```

Algorithm

Start

Declare array A[10] and variables i, j, temp

Use loop (i = 0 to 9) to read 10 integers into array A[i]

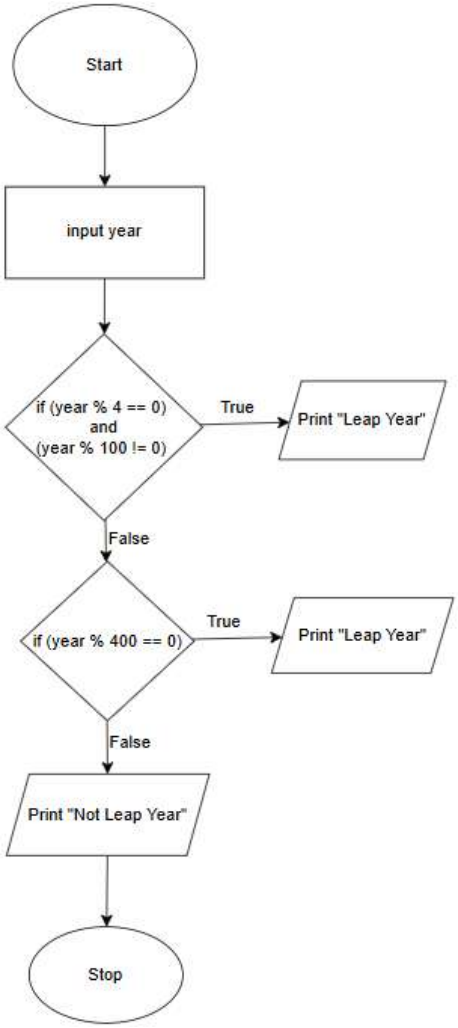
Sort using Bubble Sort:

- Repeat for i = 0 to 8:
  Loop j = 0 to 8 - i
  If A[j] > A[j+1], swap them

Print sorted array using a loop from i = 0 to 9

End

47. Draw a flow chart to check if a given year is a leapyear or not.



Program

```
#include <stdio.h>
int main() {
    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.\n", year);
    } else {
        printf("%d is Not a Leap Year.\n", year);
    }
```

```
    return 0;
}
```

Algorithm

Start

Input year

Check Conditions:

- If year % 4 == 0
  AND year % 100 != 0
  OR year % 400 == 0

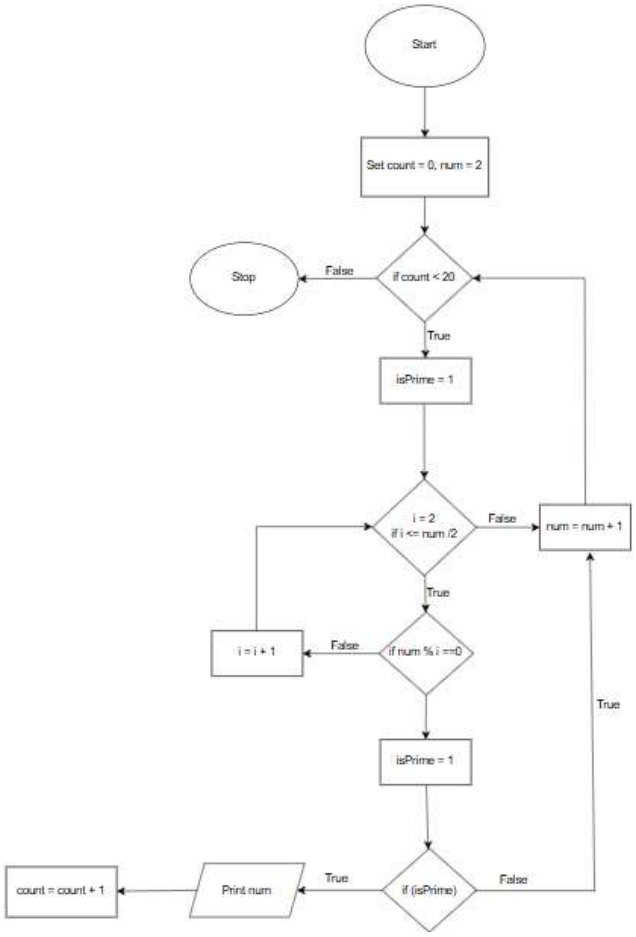
If condition is true → it's a leap year

Else → not a leap year

Print result

End

48. Draw a flow chart to print the first 20 prime numbers.



Program

```
#include <stdio.h>
int main() {
    int count = 0, num = 2, i, isPrime;
    printf("First 20 Prime Numbers:\n");
    while (count < 20) {
        isPrime = 1;
        for (i = 2; i <= num / 2; i++) {
            if (num % i == 0) {
                isPrime = 0;  // Not prime
                break;
            }
        }
        if (isPrime) {
            printf("%d ", num);
            count++;
        }
        num++;
    }
    printf("\n");
    return 0;
}
```

Algorithm

Start

Initialize: count = 0, num = 2

Repeat until count < 20:

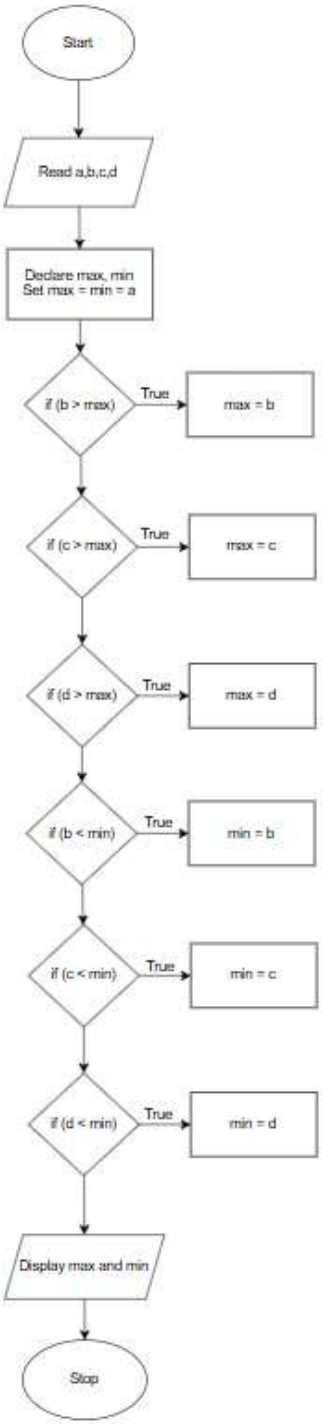
- Set isPrime = 1
- For each i = 2 to num / 2:

If num % i == 0, set isPrime = 0 and break

- If isPrime == 1:
  Print num
  Increment count
- Increment num

End

49. Draw a flow chart to find the largest number and smallest number among four given numbers.



Program

```
#include <stdio.h>
int main() {
    int a, b, c, d;
    int max, min;
    printf("Enter four numbers: ");
    scanf("%d %d %d %d", &a, &b, &c, &d);
    max = min = a;
    if (b > max) max = b;
    if (c > max) max = c;
    if (d > max) max = d;
    if (b < min) min = b;
    if (c < min) min = c;
    if (d < min) min = d;
    printf("Largest number = %d\n", max);
    printf("Smallest number = %d\n", min);
    return 0;
}
```

Algorithm

Start

Input 4 numbers: a, b, c, d

Set max = a and min = a

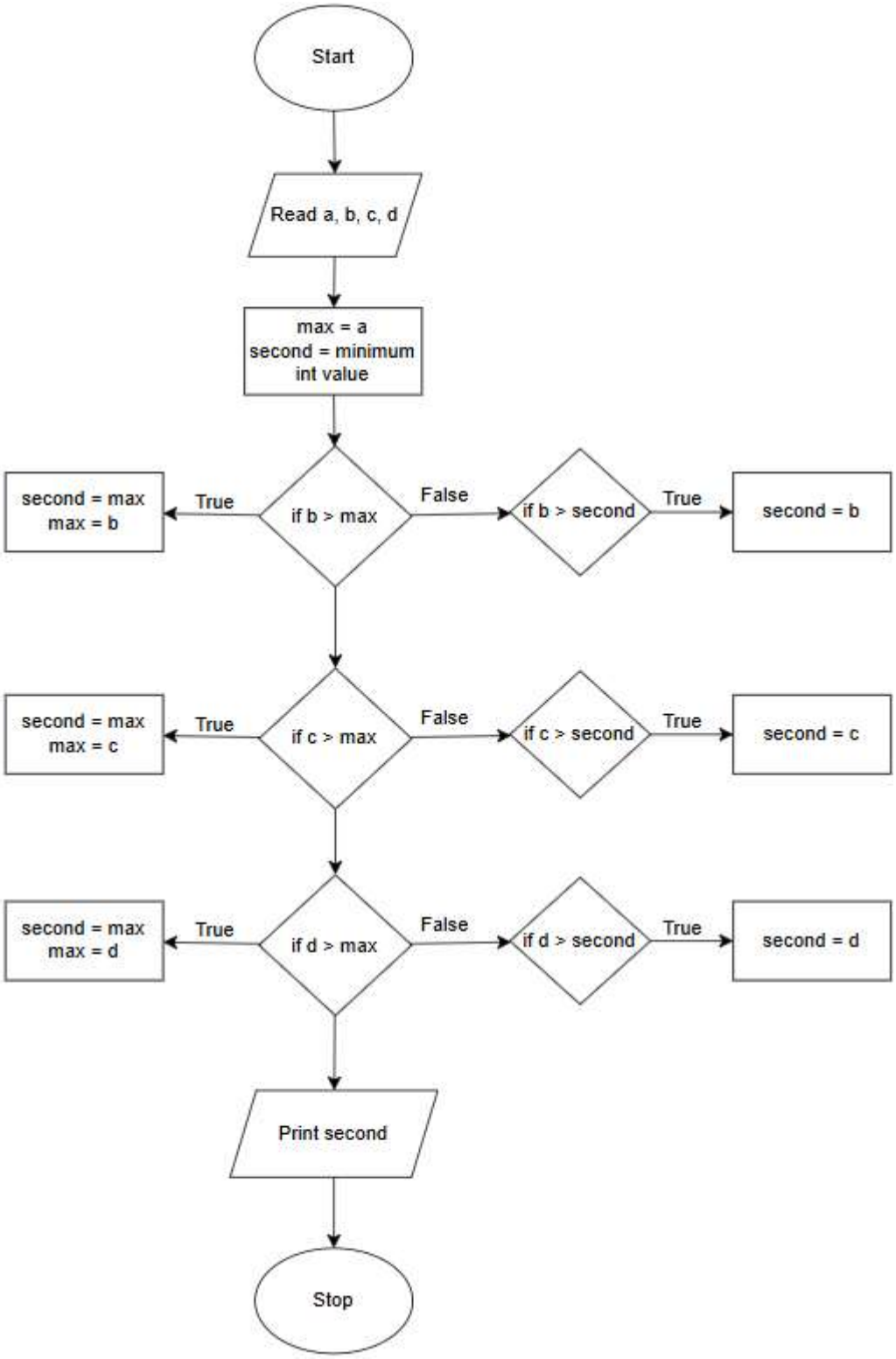
Compare b, c, and d:

- If b > max, set max = b
- If c > max, set max = c
- If d > max, set max = d
- If b < min, set min = b
- If c < min, set min = c
- If d < min, set min = d

Print max and min

Stop

50. Draw a flow chart to print second largest number among four numbers



Program

```
#include <stdio.h>
#include <limits.h>
int main() {
    int a, b, c, d;
    int max, second;
    printf("Enter four numbers: ");
    scanf("%d %d %d %d", &a, &b, &c, &d);
    max = a;
    second = INT_MIN;
    if (b > max) {
        second = max;
        max = b;
    } else if (b > second && b != max) {
        second = b;
    }
    if (c > max) {
        second = max;
        max = c;
    } else if (c > second && c != max) {
        second = c;
    }
    if (d > max) {
        second = max;
        max = d;
    } else if (d > second && d != max) {
        second = d;
    }
```

```
    printf("Second largest number = %d\n", second);
    return 0;
}
```

Algorithm

Start

Input 4 numbers: a, b, c, d

Set max = a, second = INT_MIN

Compare b:

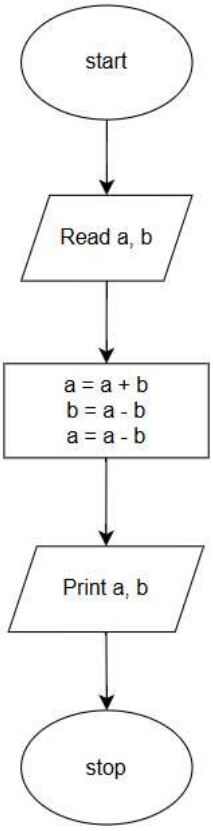
- If b > max, then:
  second = max
  max = b
- Else if b > second and b != max, then:
  second = b

Repeat same steps for c and d

After all comparisons, print second

Stop

51. Draw a flow chart to swap two numbers without using temp variable



Program

```
#include <stdio.h>
int main() {
    int a, b;
    printf("Enter two numbers: ");
    scanf("%d %d", &a, &b);
    a = a + b;
    b = a - b;
    a = a - b;
    printf("After swapping: a = %d, b = %d\n", a, b);
    return 0;
}
```

Algorithm

Start

Input two numbers a and b

a = a + b

b = a - b

a = a - b

Print a and b

Stop

52. Draw a flow chart to print given pattern

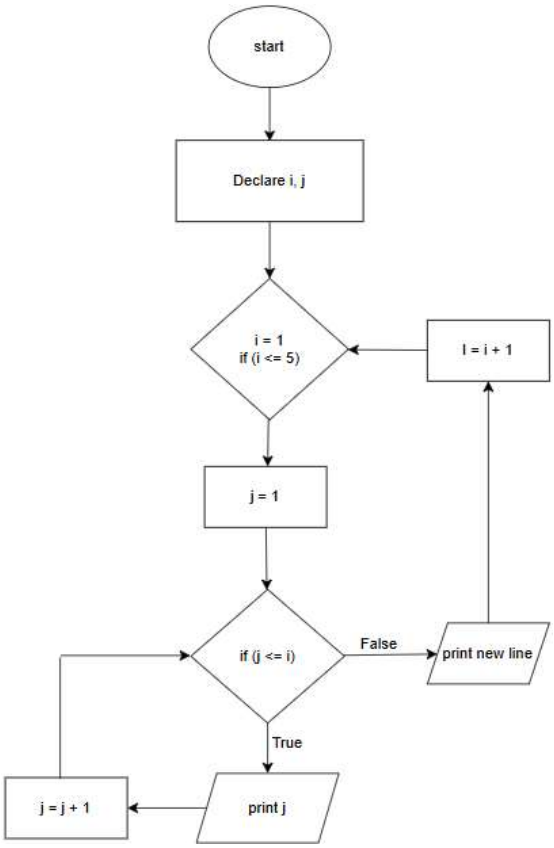
1

1 2

1 2 3

1 2 3 4

1 2 3 4 5



Program

```
#include <stdio.h>
int main() {
    int i, j;
    for (i = 1; i <= 5; i++) {
        for (j = 1; j <= i; j++) {
            printf("%d ", j);
        }
        printf("\n");
```

```
    }
    return 0;
}
```

Algorithm

Start

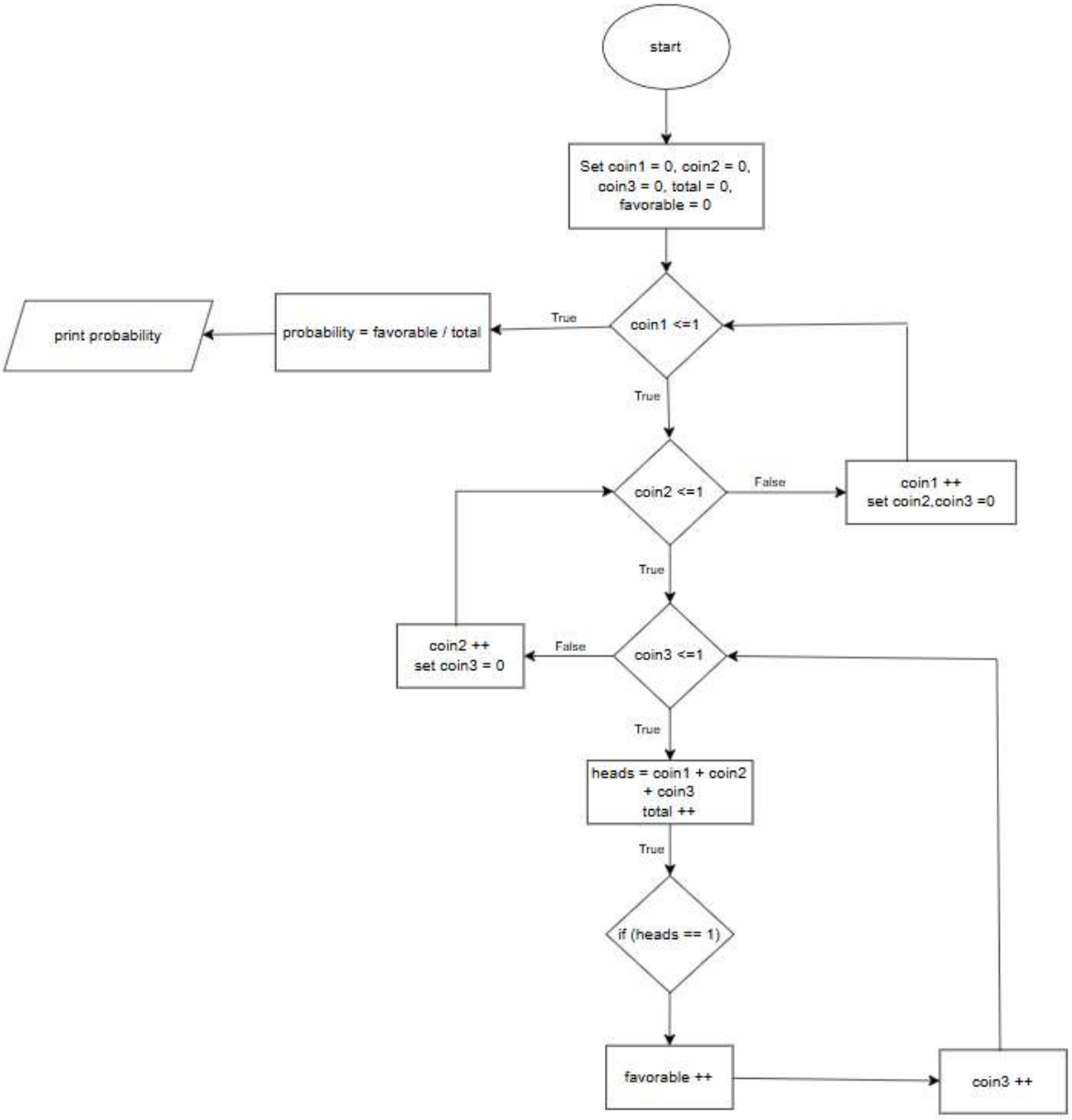
Set i = 1

Repeat while i <= 5

- Set j = 1
- While j <= i
  Print j
  Increment j
- Print newline
- Increment i

Stop

53. Draw a flow chart to find Probability of finding head when 3 coins tossed simultaneously.



Program

```
#include <stdio.h>
int main() {
    int total = 0, favorable = 0;
    int coin1, coin2, coin3;
    for (coin1 = 0; coin1 <= 1; coin1++) {
```

```
        for (coin2 = 0; coin2 <= 1; coin2++) {
            for (coin3 = 0; coin3 <= 1; coin3++) {
                int heads = coin1 + coin2 + coin3;
                total++;
                if (heads == 1) {
                    favorable++;
                }
            }
        }
    }
    float probability = (float)favorable / total;
    printf("Probability of getting exactly 1 head: %.2f\n", probability);
    return 0;
}
```

Algorithm

Start

Initialize total = 0, favorable = 0

Loop through all possible outcomes (e.g., using nested loops for 3 coin tosses)

For each outcome:

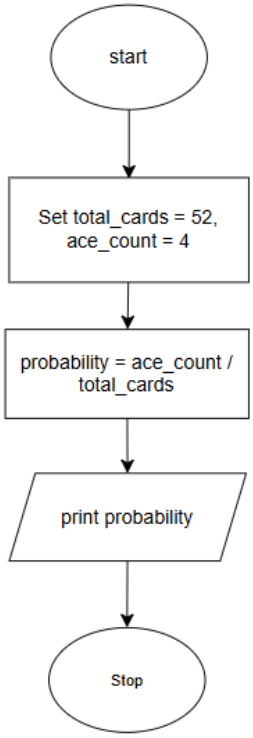
- Count number of heads
- If count == 1 → favorable++
- total++

Probability = favorable / total

Print probability

Stop

54. Draw a flow chart to find Probability of finding Ace in pack of Cards.



Program

```
#include <stdio.h>

int main() {

    int total_cards = 52;

    int ace_count = 4;

    float probability;

    probability = (float) ace_count / total_cards;

    printf("Probability of drawing an Ace = %.4f\n", probability);

    return 0;

}
```

Algorithm:

Start

Declare variables: total_cards, ace_count, probability

Assign:

- total_cards = 52
- ace_count = 4

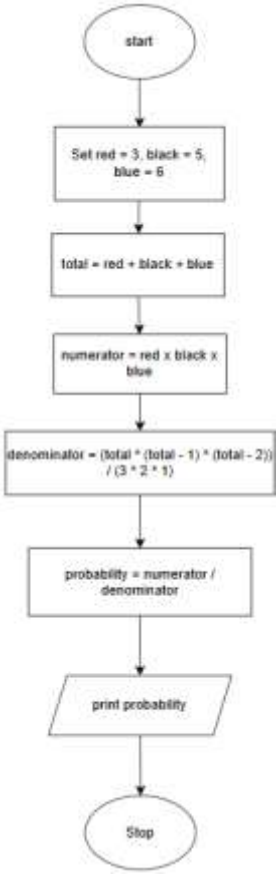
Calculate:

- probability ← ace_count / total_cards

Print probability

Stop

55. In a box we have 3 red, 5 black, 6 blue balls, if we draw 3 balls simultaneously. Draw a flow chart to find the probability of getting 1 red, 1 blue, 1 black?



Program

```
#include <stdio.h>

int main() {
    int red = 3, black = 5, blue = 6;
    int total = red + black + blue;
    int numerator = red * black * blue;
```

```
    int denominator = (total * (total - 1) * (total - 2)) / (3 * 2 * 1);
    float probability = (float) numerator / denominator;

    printf("Probability of drawing 1 Red, 1 Blue, 1 Black = %.4f\n", probability);
    return 0;
}
```

Algorithm

Start

Initialize values:

- red = 3
- black = 5
- blue = 6
- total = red + black + blue = 14

Calculate numerator:

- numerator = red × black × blue = 3 × 5 × 6 = 90

Calculate denominator:

- denominator = C(14, 3) = (14 × 13 × 12) / (3 × 2 × 1) = 364

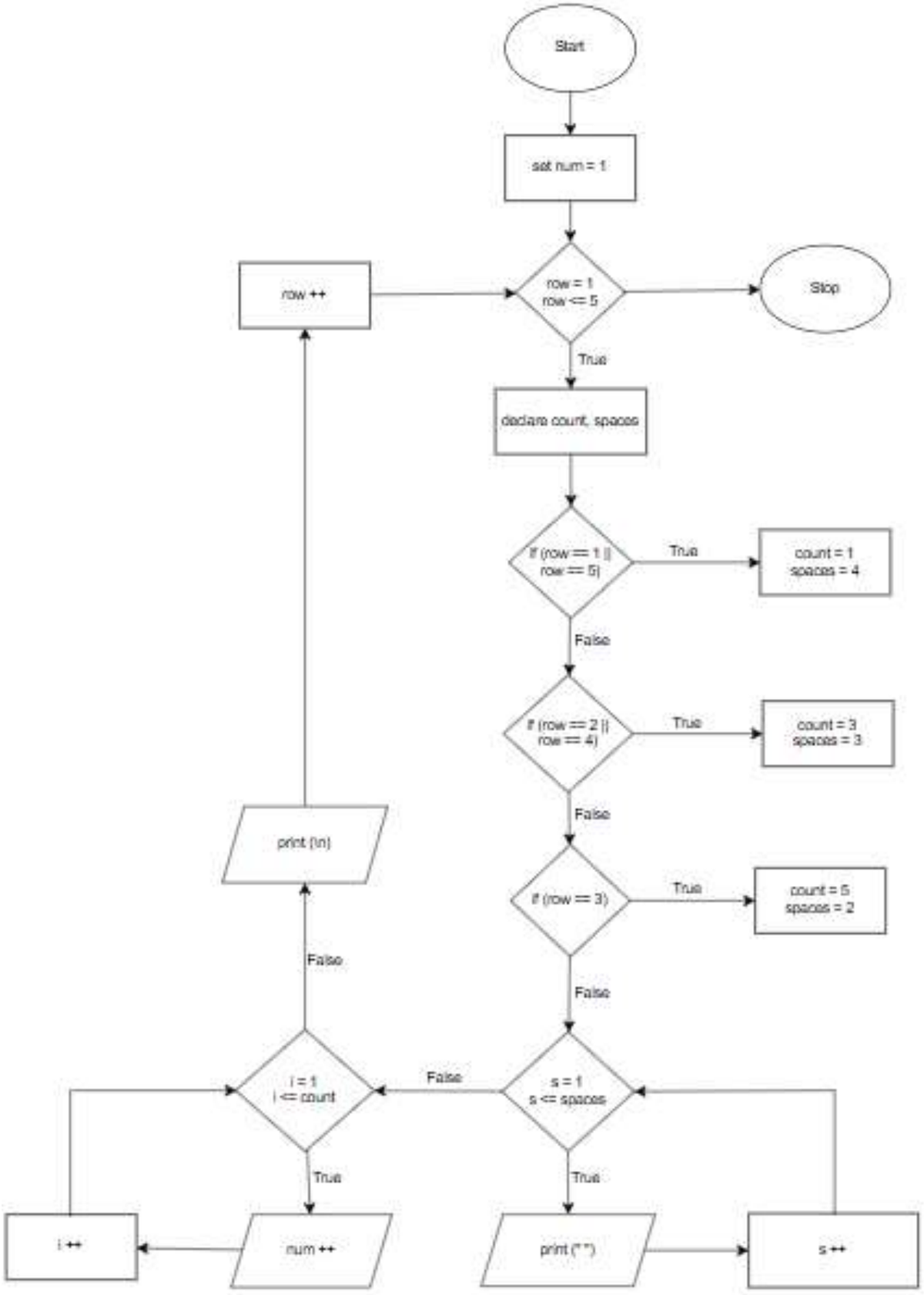
Probability = numerator / denominator

Print probability

Stop

56. Draw a flow chart to print given pattern

```
      1
   2  3  4
5  6  7  8  9
 10 11 12
    13
```


Start
set num = 1
row ++
row = 1
row <= 5
Stop
True
declare count, spaces
if (row == 1 ||
row == 5)
True
count = 1
spaces = 4
False
if (row == 2 ||
row == 4)
True
count = 3
spaces = 3
False
print (\n)
if (row == 3)
True
count = 5
spaces = 2
False
False
i = 1
i <= count
False
s = 1
s <= spaces
True
True
i ++
num ++
print (" ")
s ++

Program

```
#include <stdio.h>
int main() {
    int num = 1;
    for (int row = 1; row <= 5; row++) {
        int count, spaces;
        if (row == 1 || row == 5) {
            count = 1;
            spaces = 4;
        } else if (row == 2 || row == 4) {
            count = 3;
            spaces = 3;
        } else if (row == 3) {
            count = 5;
            spaces = 2;
        }
        for (int s = 1; s <= spaces; s++) {
            printf("  ");
        }
        for (int i = 1; i <= count; i++) {
            printf("%2d ", num++);
        }
        printf("\n");
    }
    return 0;
}
```

Algorithm:

1. Start
2. Initialize:
    - num = 1
3. Loop from row = 1 to 5
    - Determine spaces based on row
    - Determine count (how many numbers to print)
    - Print spaces
    - Print count numbers starting from num
    - Increment num for each number printed
4. Stop

57. Draw a flow chart to print given pattern

*** * * * ***

*** * * ***

*** * ***

*** ***

*****

Algorithm:

Start

Initialize i = 5

Loop while i >= 1

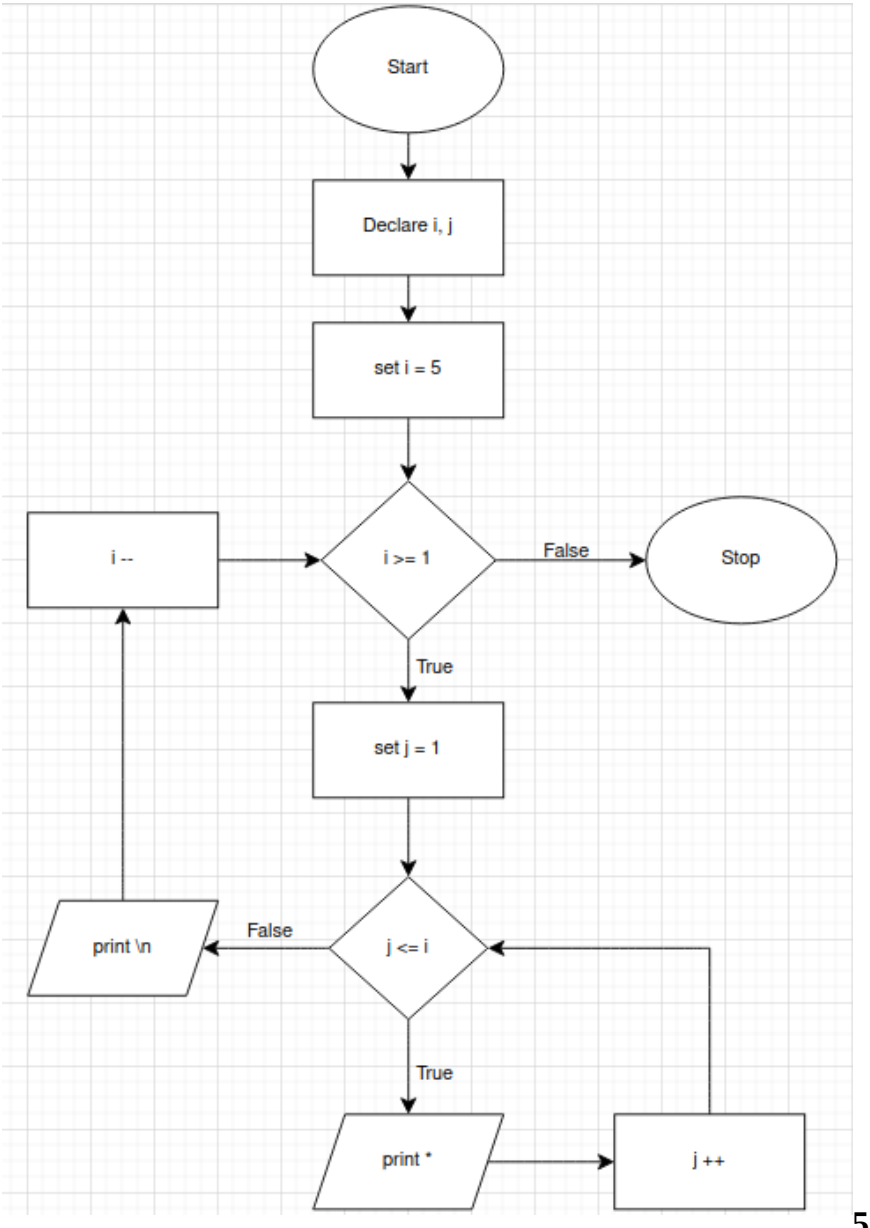
- Initialize j = 1
- Loop while j <= i
    Print *
    Increment j
- Print newline
- Decrement i

End

Program

#include <stdio.h>

```
int main() {
    int i, j;
    for (i = 5; i >= 1; i--) {
        for (j = 1; j <= i; j++) {
            printf("* ");
        }
        printf("\n");
    }
    return 0;
}
```



58. Draw a flow chart to print given pattern

*** * * * ***

*** * * ***

*** * ***

*** ***

*****

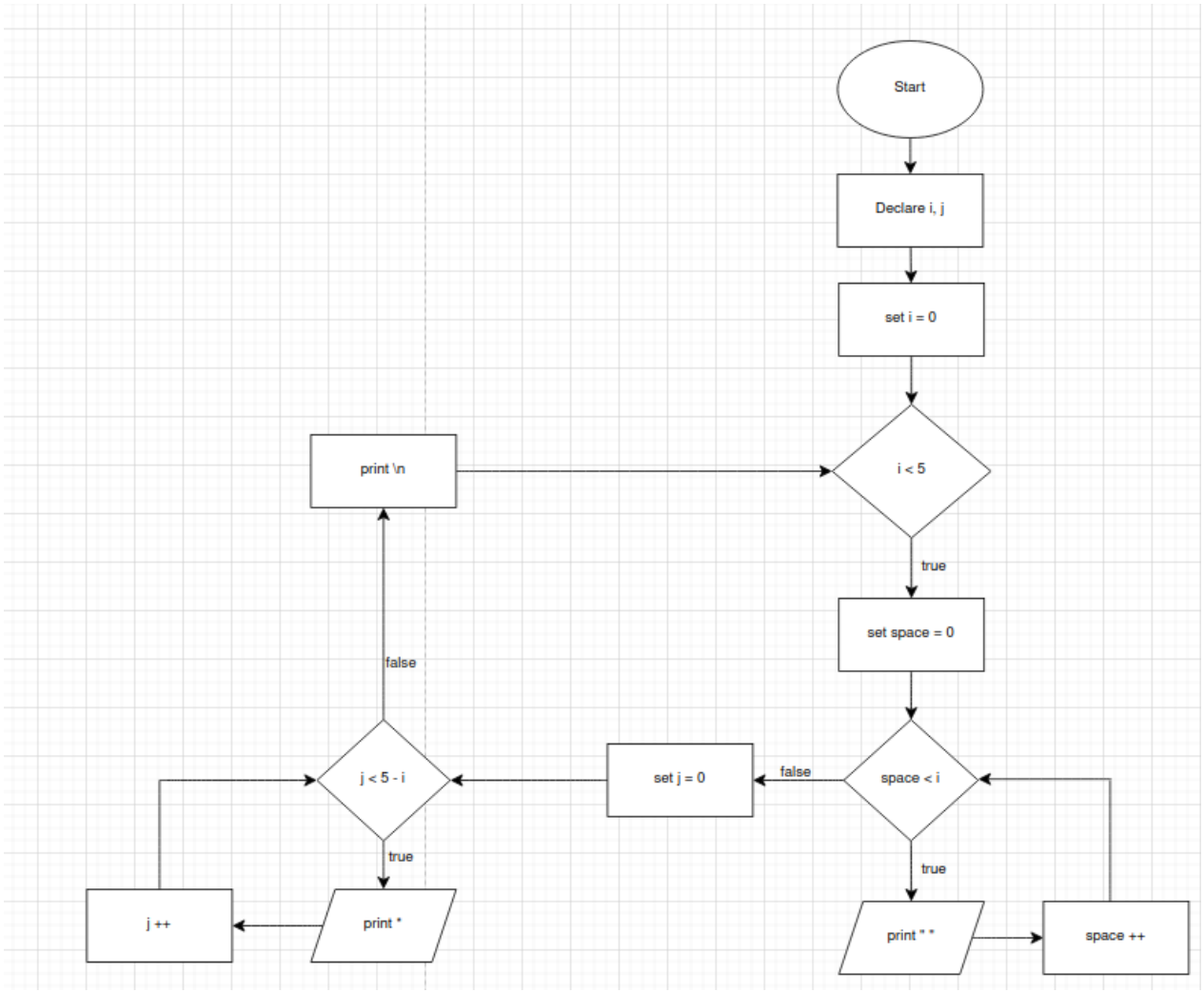
Algorithm:

1. Start
2. Initialize i = 5

3. Repeat while i >= 1
    - Initialize j = 1
    - Repeat while j <= i
        - Print *
        - Increment j
    - Print newline
    - Decrement i
4. End

Program

```
#include <stdio.h>
int main() {
   int i, j;
   for (i = 5; i >= 1; i--) {
      for (j = 1; j <= i; j++) {
         printf("* ");
      }
      printf("\n");
   }
   return 0;
}
```



59. Draw a flow chart to print given pattern

```
        *
      * * *
    * * * * *
  * * * * * * *
```

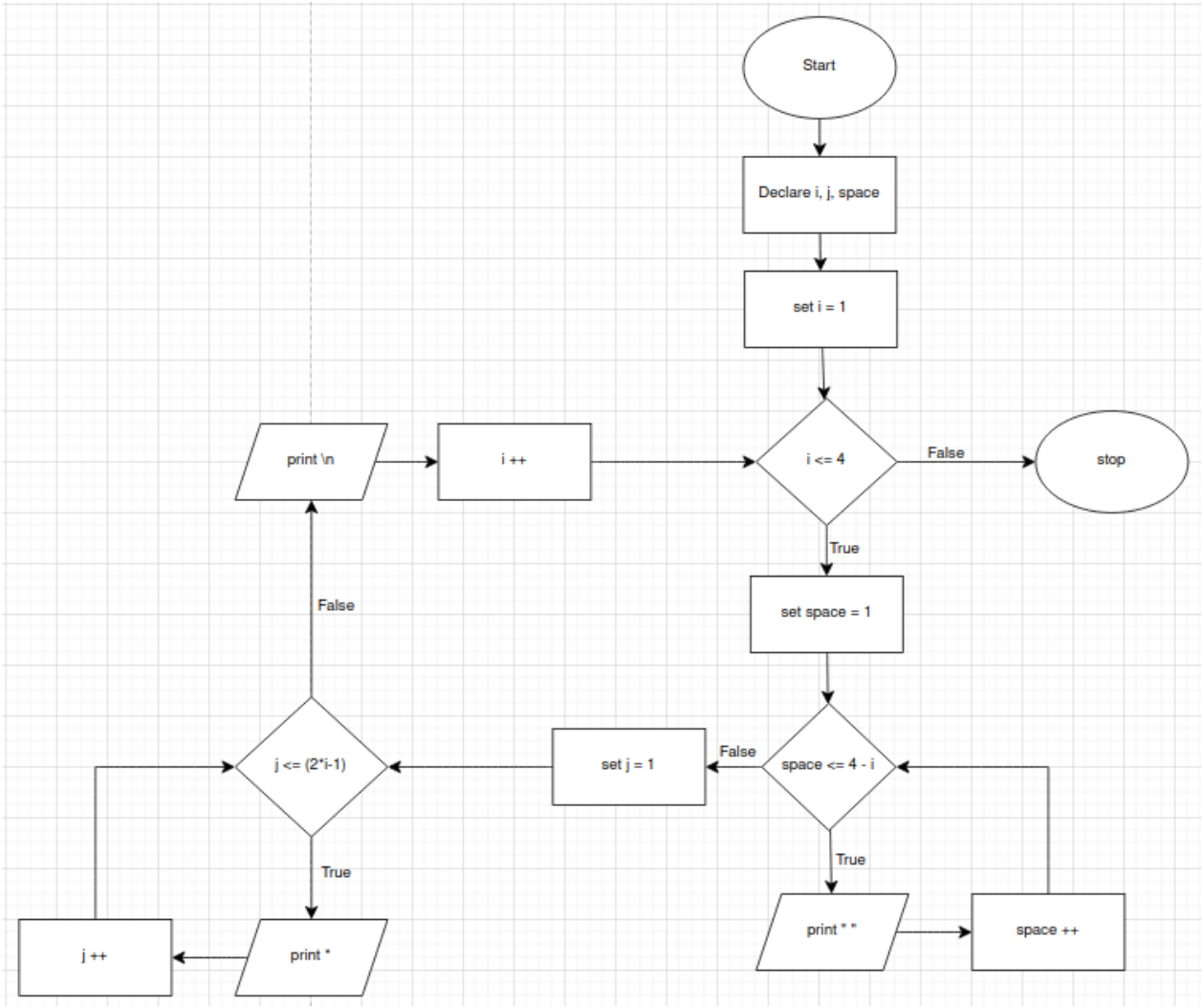
Algorithm:

1. Start
2. Set i = 1
3. Repeat while i <= 4
    - Set s = 1
    - Repeat while s <= (4 - i)
        - Print space " " (double-space for alignment)

        - Increment s
    - Set j = 1
    - Repeat while j <= (2 * i - 1)
        - Print "* "
        - Increment j
    - Print newline
    - Increment i
4. End

Program

```
#include <stdio.h>
int main() {
    int i, j, space;
    for (i = 1; i <= 4; i++) {
        for (space = 1; space <= 4 - i; space++) {
            printf("  ");
        }
        for (j = 1; j <= (2 * i - 1); j++) {
            printf("* ");
        }
        printf("\n");
    }
    return 0;
}
```



60.Draw a flow chart to print given pattern

```
* * * * * * *
  * * * * *
    * * *
      *
```

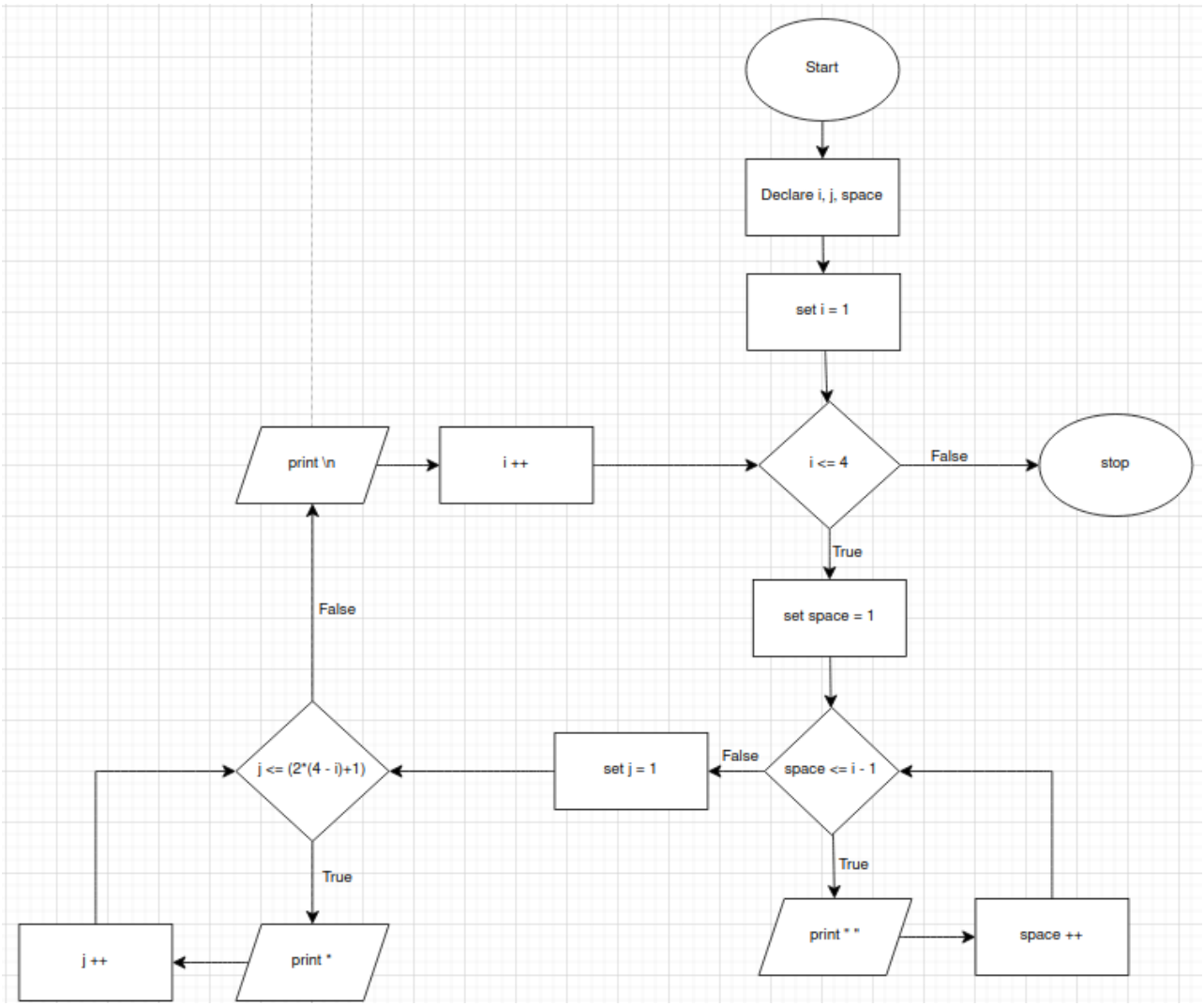
Algorithm

1. Start
2. Set i = 1
3. Repeat while i <= 4:
    - Set s = 1
    - Repeat while s <= (i - 1)
        - Print " " (2 spaces for alignment)

        - Increment s
    - Set j = 1
    - Repeat while j <= 2 * (4 - i) + 1:
        - Print "* "
        - Increment j
    - Print newline
    - Increment i
4. Stop

Program

```
#include <stdio.h>
int main() {
    int i, j, space;
    for (i = 1; i <= 4; i++) {
        for (space = 1; space <= i - 1; space++) {
            printf("  ");
        }
        for (j = 1; j <= 2 * (4 - i) + 1; j++) {
            printf("* ");
        }
        printf("\n");
    }
    return 0;
}
```



61. Draw a flow chart to find roots of a quadratic equation?

Algorithm:

1. Start
2. Input values for a, b, and c
3. Compute the discriminant: $D = b^2 - 4ac$
4. Check the nature of D:
    - If $D > 0$:
      Roots are real and distinct
      Use:
    - Else if $D == 0$:
      Roots are real and equal
      Use:

- $$x_1 = \frac{-b + \sqrt{D}}{2a}, \quad x_2 = \frac{-b - \sqrt{D}}{2a}$$
- Else:
  Roots are complex
  Use:

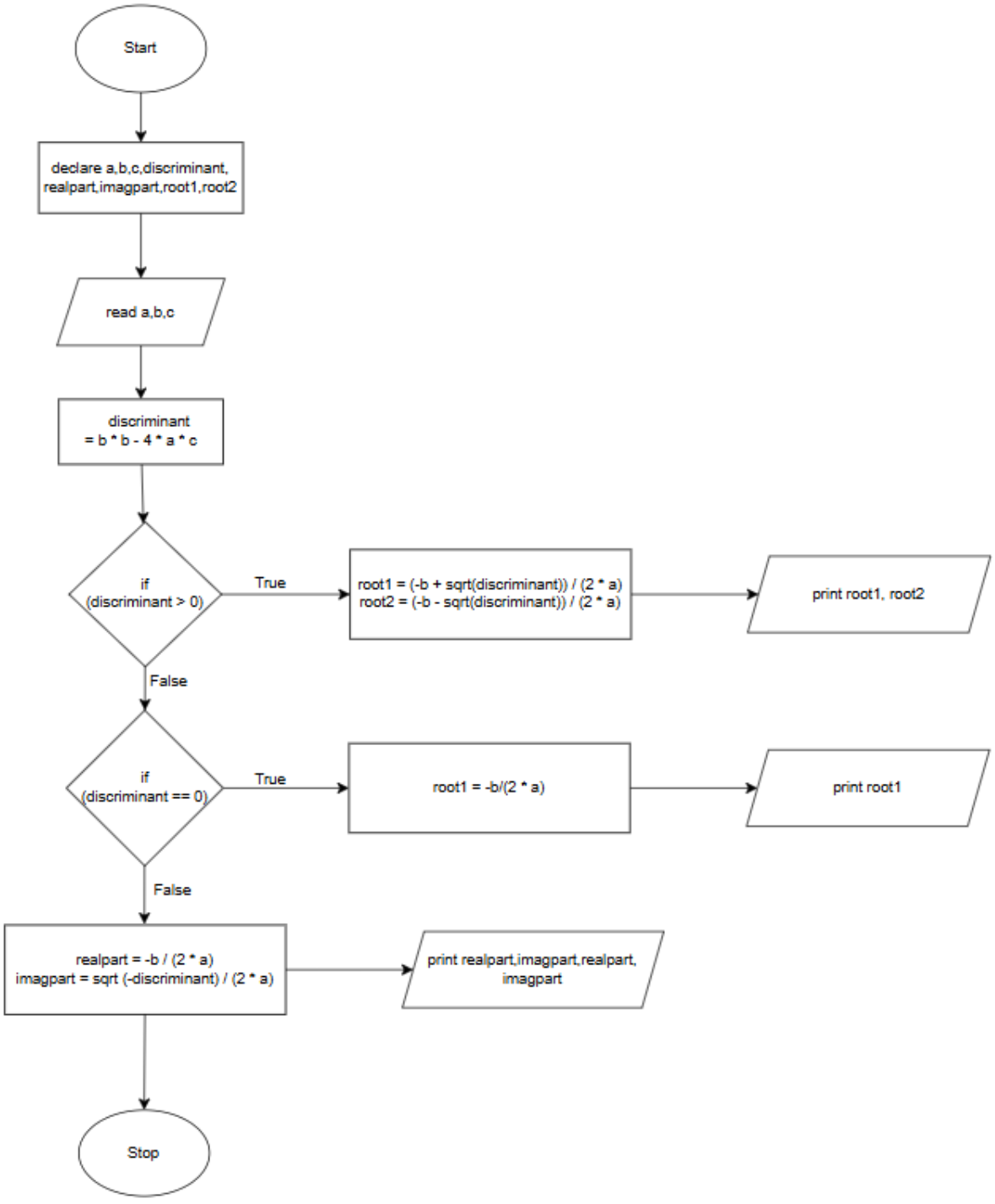
$$\text{Real Part} = \frac{-b}{2a}, \quad \text{Imaginary Part} = \frac{\sqrt{-D}}{2a}$$

5. Display the roots

Program

```
#include <stdio.h>
#include <math.h>
int main() {
    float a, b, c, discriminant, realPart, imagPart, root1, root2;
    printf("Enter coefficients a, b and c: ");
    scanf("%f %f %f", &a, &b, &c);
    discriminant = b * b - 4 * a * c;
    if (discriminant > 0) {
        root1 = (-b + sqrt(discriminant)) / (2 * a);
        root2 = (-b - sqrt(discriminant)) / (2 * a);
        printf("Roots are real and distinct: %.2f and %.2f\n", root1, root2);
    } else if (discriminant == 0) {
        root1 = -b / (2 * a);
        printf("Roots are real and equal: %.2f\n", root1);
    } else {
        realPart = -b / (2 * a);
        imagPart = sqrt(-discriminant) / (2 * a);
        printf("Roots are complex: %.2f + %.2fi and %.2f - %.2fi\n",
               realPart, imagPart, realPart, imagPart);
```

```
    }
    return 0;
}
```


Start
declare a,b,c,discriminant, realpart,imagpart,root1,root2
read a,b,c
discriminant = b * b - 4 * a * c
if (discriminant > 0)
True
root1 = (-b + sqrt(discriminant)) / (2 * a)
root2 = (-b - sqrt(discriminant)) / (2 * a)
print root1, root2
False
if (discriminant == 0)
True
root1 = -b/(2 * a)
print root1
False
realpart = -b / (2 * a)
imagpart = sqrt (-discriminant) / (2 * a)
print realpart,imagpart,realpart, imagpart
Stop

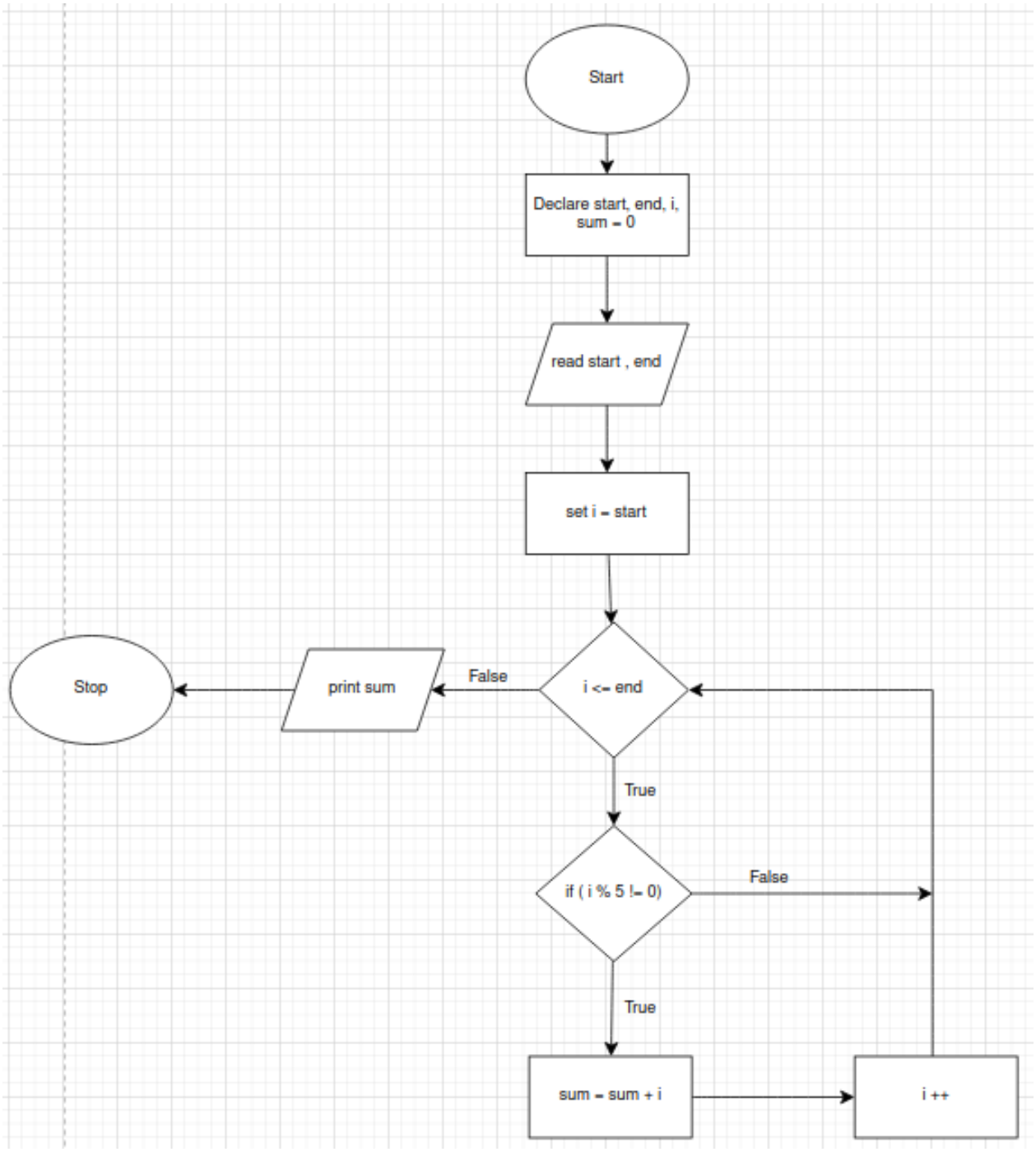
62. Take two numbers from the user and draw a flow chart to find the sum of numbers which are not divisible by 5 between the ranges.

Algorithm:

1. Start
2. Input start and end
3. Initialize sum = 0
4. Repeat from i = start to end:
    - If i % 5 != 0:
        - sum = sum + i
5. Print sum
6. End

Program

```
#include <stdio.h>
int main() {
   int start, end, i, sum = 0;
   printf("Enter start of range: ");
   scanf("%d", &start);
   printf("Enter end of range: ");
   scanf("%d", &end);
   for (i = start; i <= end; i++) {
      if (i % 5 != 0) {
         sum += i;
      }
   }
   printf("Sum of numbers not divisible by 5 = %d\n", sum);
   return 0;
}
```


Start
Declare start, end, i, sum = 0
read start , end
set i = start
i <= end
False
print sum
Stop
True
if (i % 5 != 0)
False
True
sum = sum + i
i ++

63. Draw a flow chart to print given pattern

Hollow square

```
# # # # #
#       #
#       #
#       #
# # # # #
```

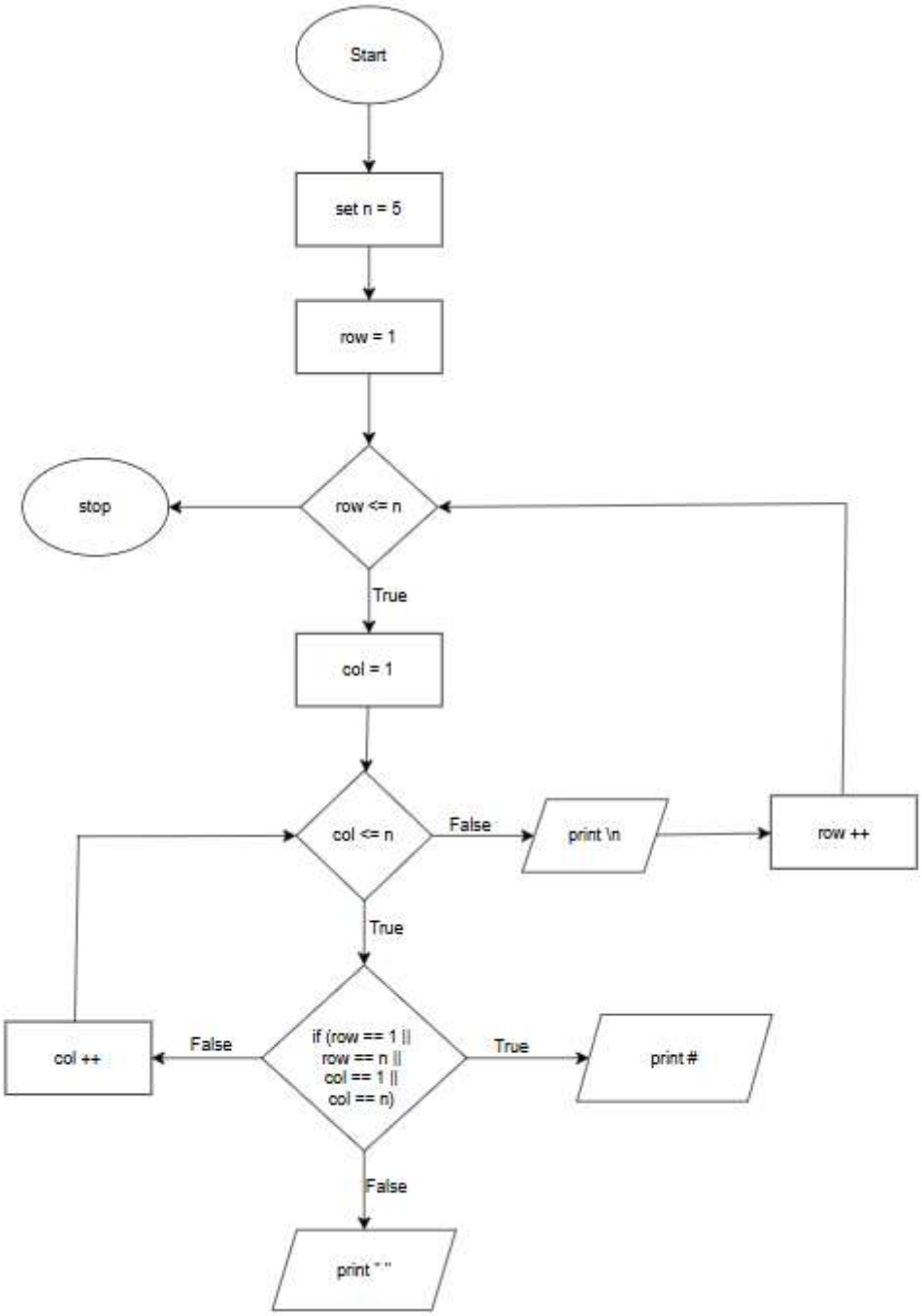
Algorithm:

1. Start
2. Set n = 5 (side length of the square)
3. Loop from row = 1 to n
    - Loop from col = 1 to n
        - If row == 1 || row == n || col == 1 || col == n
          → Print "# "
        - Else
          → Print " " (two spaces)
    - Print newline
4. End

Program

```
#include <stdio.h>
int main() {
   int n = 5;
   for (int row = 1; row <= n; row++) {
      for (int col = 1; col <= n; col++) {
         if (row == 1 || row == n || col == 1 || col == n)
            printf("# ");
         else
            printf("  ");
```

```
        }
        printf("\n");
    }
    return 0;
}
```



64. Draw a flow chart to find Armstrong numbers between two intervals (user input)?

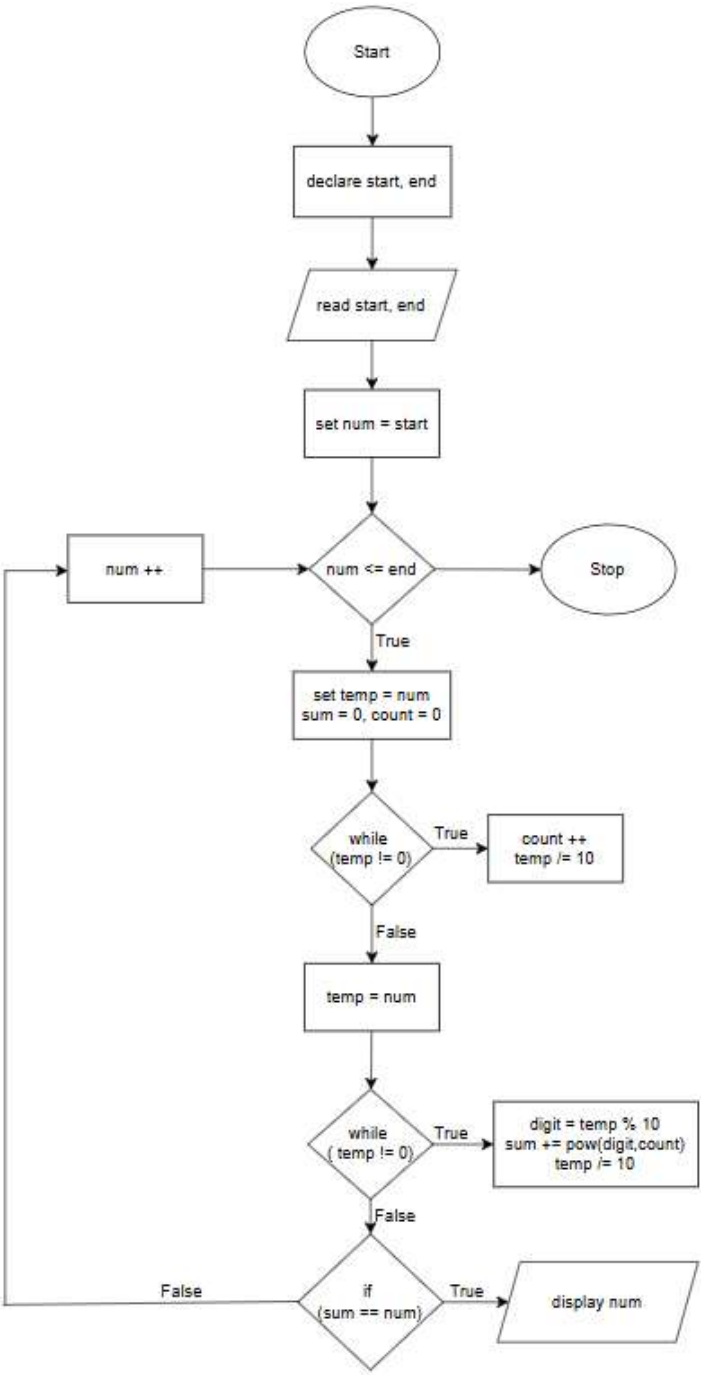
Algorithm:

1. **Start**
2. Input start, end
3. Loop from num = start to end:
    1. Set sum = 0, temp = num, count = 0
    2. Count number of digits in temp
        - While temp > 0:
            - count++
            - temp = temp / 10
    3. Set temp = num, sum = 0
    4. While temp > 0:
        - digit = temp % 10
        - sum = sum + (digit^count)
        - temp = temp / 10
    5. If sum == num, print num
4. End

Program:

```
#include <stdio.h>
#include <math.h>
int main() {
    int start, end;
    printf("Enter start and end values: ");
    scanf("%d %d", &start, &end);
    for (int num = start; num <= end; num++) {
        int temp = num, sum = 0, count = 0;
        while (temp != 0) {
            count++;
```

```
            temp /= 10;
        }
        temp = num;
        while (temp != 0) {
            int digit = temp % 10;
            sum += pow(digit, count);
            temp /= 10;
        }
        if (sum == num) {
            printf("%d is an Armstrong number\n", num);
        }
    }
    return 0;
}
```



65. Draw a flow chart to convert binary number to octal.

Algorithm:

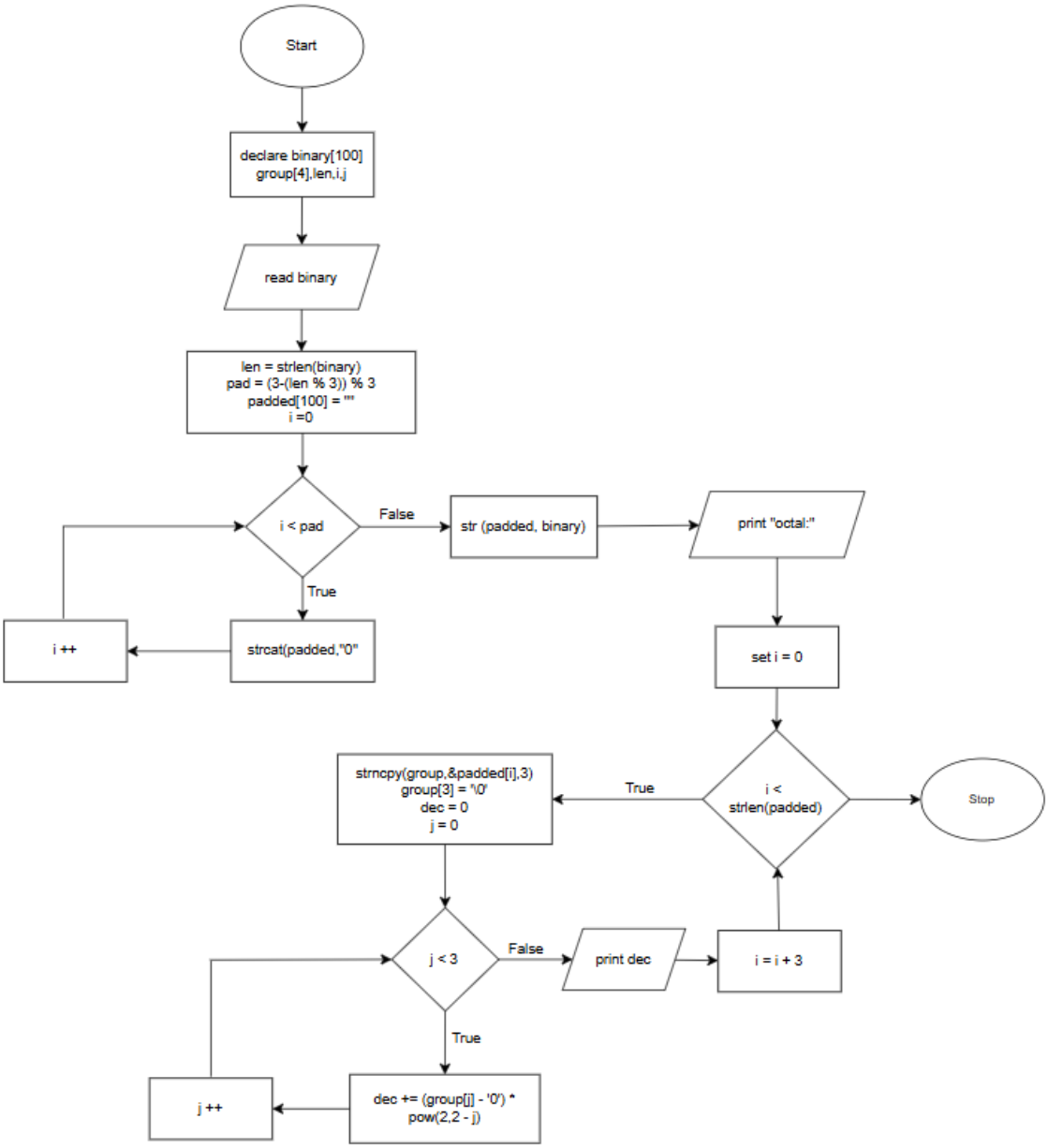
1. **Start**
2. Input binary number as a string (or integer array)
3. Pad the binary number from the left with zeros so that the length is a multiple of 3
4. Set index i = 0

5. Repeat until end of binary string:
    - Take 3 binary digits
    - Convert the 3-bit binary group to decimal (this becomes one octal digit)
    - Append the result to the octal number
6. Print the octal number
7. End

Program:

```
#include <stdio.h>
#include <string.h>
#include <math.h>
int main() {
    char binary[100], group[4];
    int len, i, j;
    printf("Enter binary number: ");
    scanf("%s", binary);
    len = strlen(binary);
    int pad = (3 - (len % 3)) % 3;
    char padded[100] = "";
    for (i = 0; i < pad; i++)
        strcat(padded, "0");
    strcat(padded, binary);
    printf("Octal: ");
    for (i = 0; i < strlen(padded); i += 3) {
        strncpy(group, &padded[i], 3);
        group[3] = '\0';
        int dec = 0;
        for (j = 0; j < 3; j++) {
            dec += (group[j] - '0') * pow(2, 2 - j);
        }
```

```
        printf("%d", dec);
    }
    printf("\n");
    return 0;
}
```



66. Draw a flow chart to convert binary number to hexa decimal.

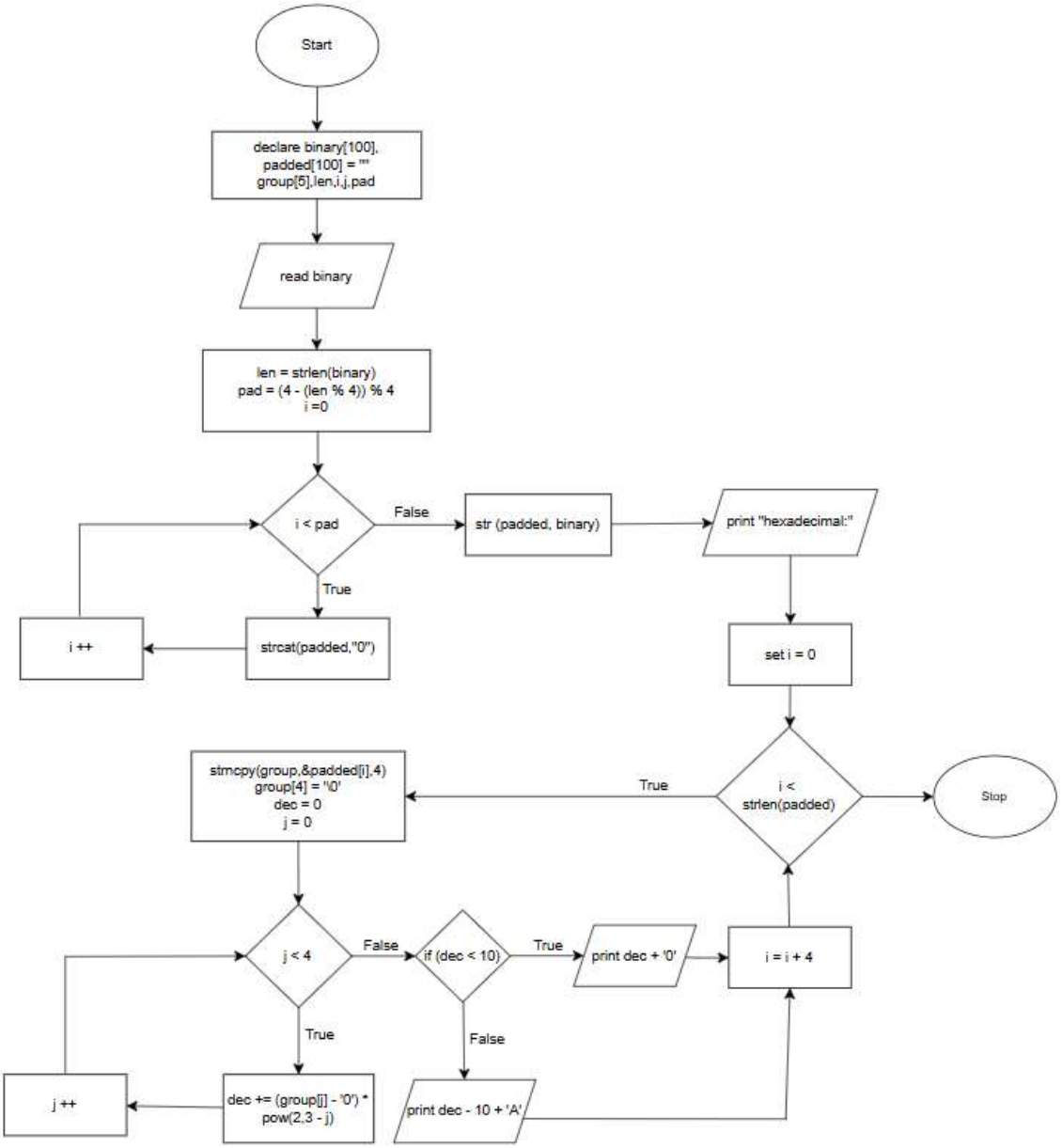
Algorithm:

1. Start
2. Input binary number as string
3. Pad binary from the left with zeros to make its length a multiple of 4
4. Initialize i = 0
5. Repeat until end of binary:
    - Take 4 bits
    - Convert 4-bit binary group to decimal
    - Convert decimal to hexadecimal digit
    - Append to result
6. Print hexadecimal result
7. End

Program

```
#include <stdio.h>
#include <string.h>
#include <math.h>
int main() {
   char binary[100], padded[100] = "", group[5];
   int len, i, j, pad;
   printf("Enter binary number: ");
   scanf("%s", binary);
   len = strlen(binary);
   pad = (4 - (len % 4)) % 4;
   for (i = 0; i < pad; i++)
      strcat(padded, "0");
   strcat(padded, binary);
   printf("Hexadecimal: ");
```

```
    for (i = 0; i < strlen(padded); i += 4) {
        strncpy(group, &padded[i], 4);
        group[4] = '\0';
        int dec = 0;
        for (j = 0; j < 4; j++) {
            dec += (group[j] - '0') * pow(2, 3 - j);
        }
        if (dec < 10)
            printf("%c", dec + '0');
        else
            printf("%c", dec - 10 + 'A');
    }
    printf("\n");
    return 0;
}
```



67. Draw a flow chart to print given pattern .

Pascals triangle

1

1 1

1 2 2 1

1 3 4 3 1

1 4 7 7 4 1

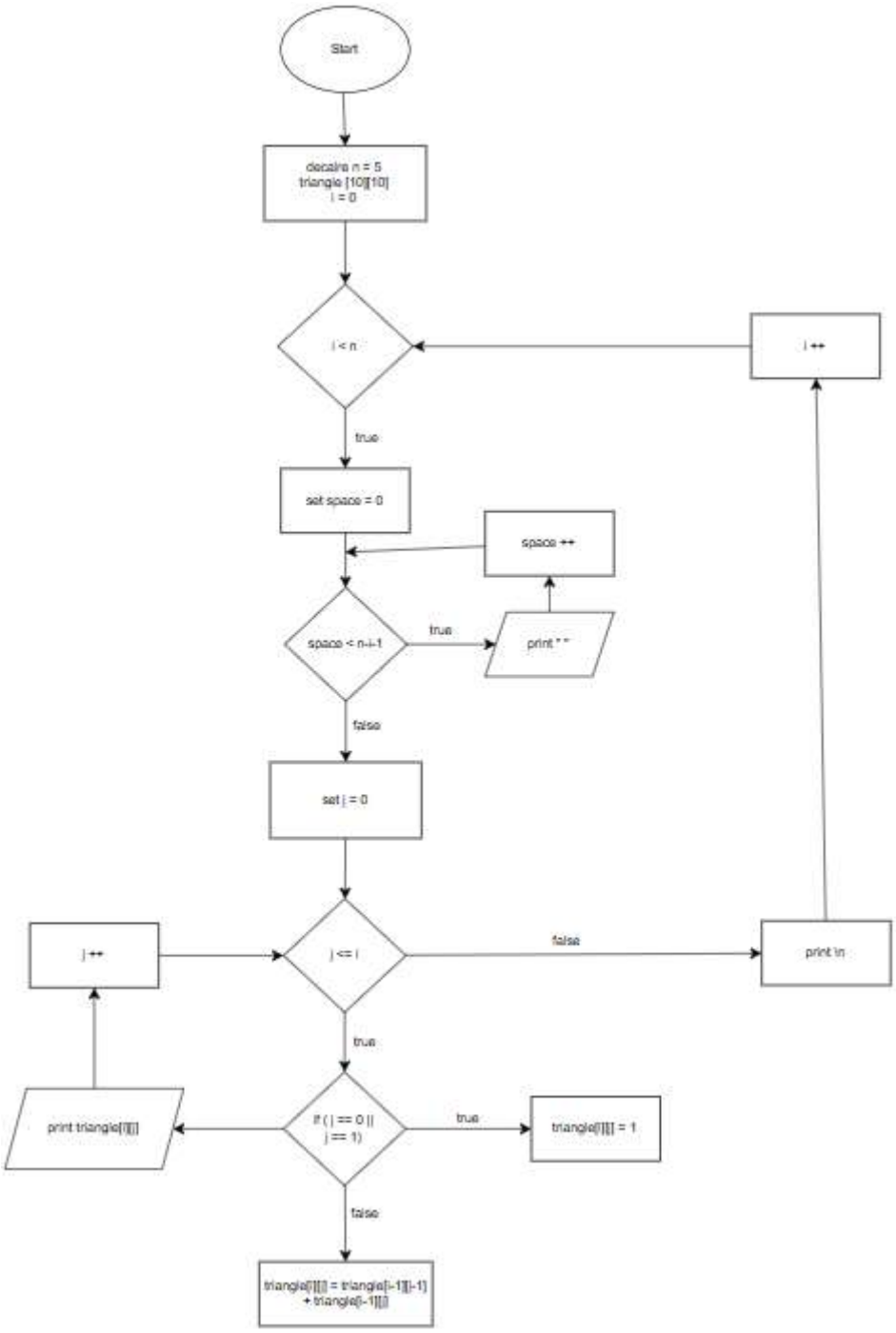
Algorithm:

1. Start

2. Input number of rows (say n = 5)
3. Loop i = 0 to n-1:
    - Print spaces → (n - i - 1) times for center alignment
    - Loop j = 0 to i:
        - If j == 0 || j == i → val = 1
        - Else: val = triangle[i-1][j-1] + triangle[i-1][j]
        - Print val
4. Move to next line
5. End

Program

```
#include <stdio.h>
int main() {
    int n = 5;
    int triangle[10][10];
    for (int i = 0; i < n; i++) {
        for (int space = 0; space < n - i - 1; space++) {
            printf("  ");
        }
        for (int j = 0; j <= i; j++) {
            if (j == 0 || j == i)
                triangle[i][j] = 1;
            else
                triangle[i][j] = triangle[i-1][j-1] + triangle[i-1][j];
            printf("%d ", triangle[i][j]);
        }
        printf("\n");
    }
    return 0;
}
```



68. Draw a flow chart to take a string from the user and print all the combinations

Algorithm (Backtracking Approach):

1. Start
2. Input a string str
3. Call recursive function permute(str, left, right)
4. In permute():

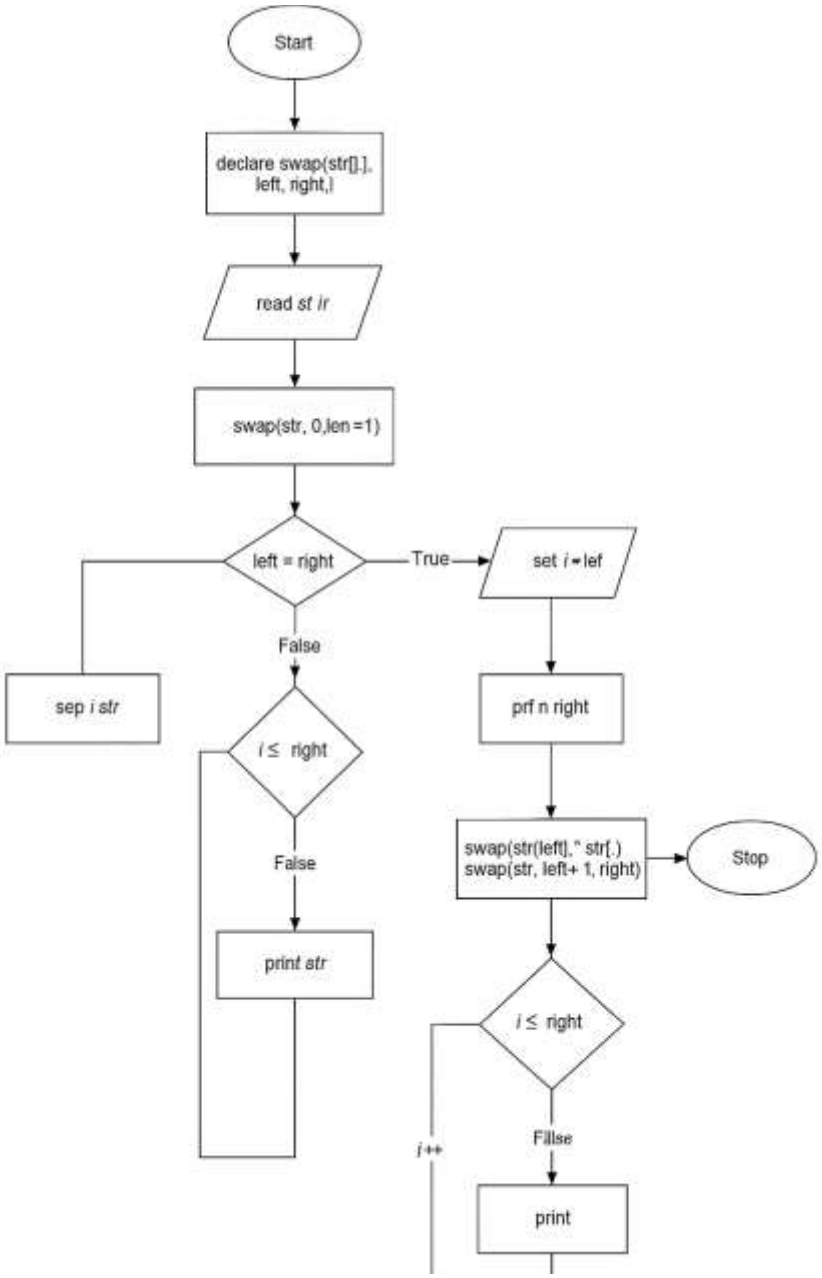
- If left == right: print str
- Else:
  - Loop from i = left to right:
    - Swap str[left] with str[i]
    - Call permute(str, left + 1, right)
    - Backtrack: Swap back str[left] with str[i]

5. End

Program

```
#include <stdio.h>
#include <string.h>
void swap(char *x, char *y) {
   char temp = *x;
   *x = *y;
   *y = temp;
}
void permute(char *str, int left, int right) {
   if (left == right) {
      printf("%s\n", str);
      return;
   }
   for (int i = left; i <= right; i++) {
      swap(&str[left], &str[i]);
      permute(str, left + 1, right);
      swap(&str[left], &str[i]); // backtrack
   }
}
int main() {
   char str[100];
   printf("Enter a string: ");
```

```
    scanf("%s", str);
    int len = strlen(str);
    permute(str, 0, len - 1);
    return 0;
}
```



69.Draw a flow chart to print given pattern

1

2 3 2

3 4 5 4 3

4 5 6 7 6 5 4

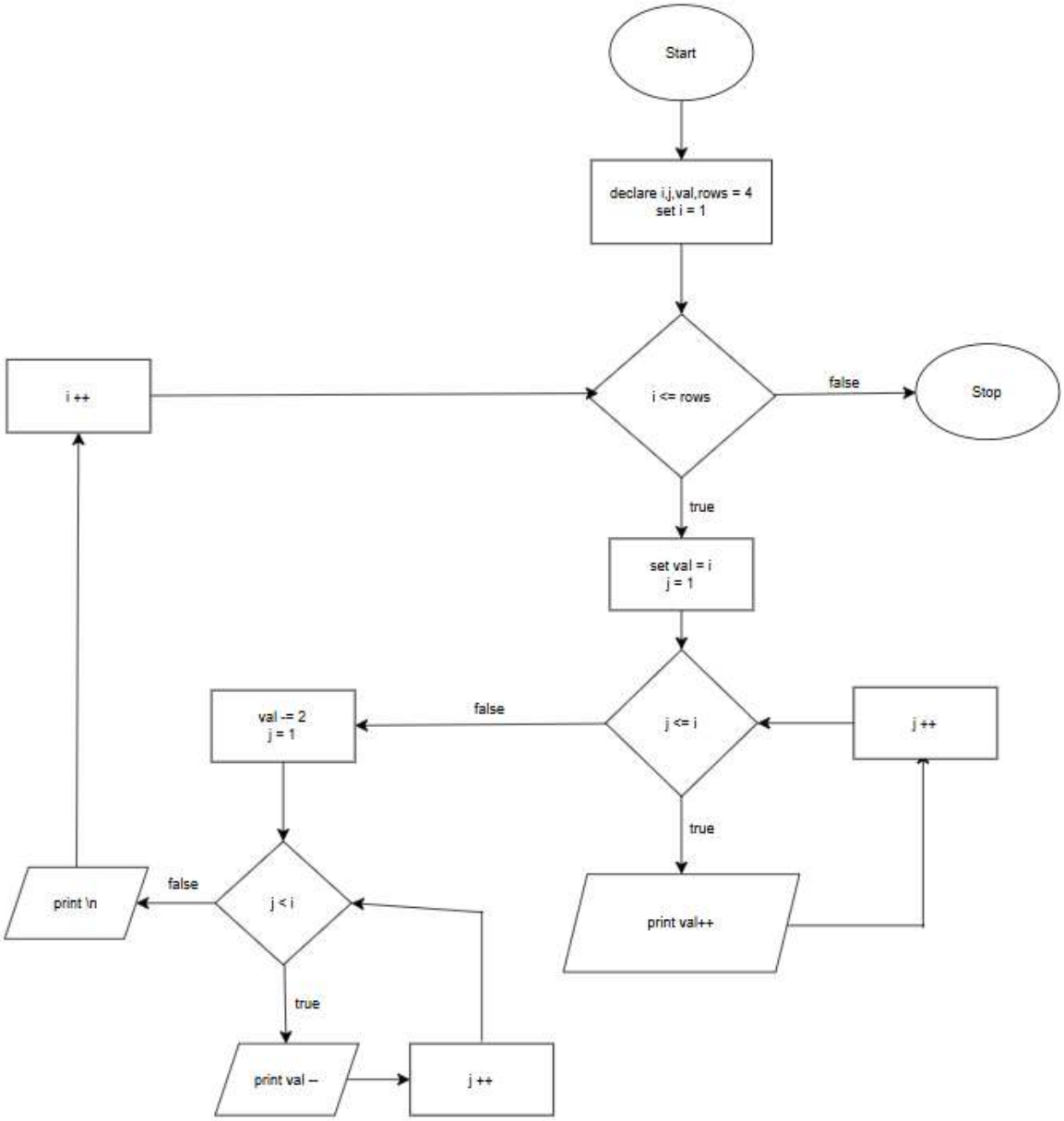
Algorithm:

1. Start
2. Set rows = 4
3. Loop `i = 1` to `rows`
    - Set `val = i`
    - Loop `j = 1` to `i` → Print `val++`
    - Set `val = val - 2`
    - Loop `j = 1` to `i - 1` → Print `val--`
    - Print newline
4. End

program

```
#include <stdio.h>
int main() {
   int i, j, val, rows = 4;
   for (i = 1; i <= rows; i++) {
      val = i;
      for (j = 1; j <= i; j++) {
         printf("%d ", val++);
      }
      val -= 2;
      for (j = 1; j < i; j++) {
         printf("%d ", val--);
      }
      printf("\n");
   }
   return 0;
```

}



70.Draw a flow chart to find no. of vowels and consonants in a string?

Algorithm:

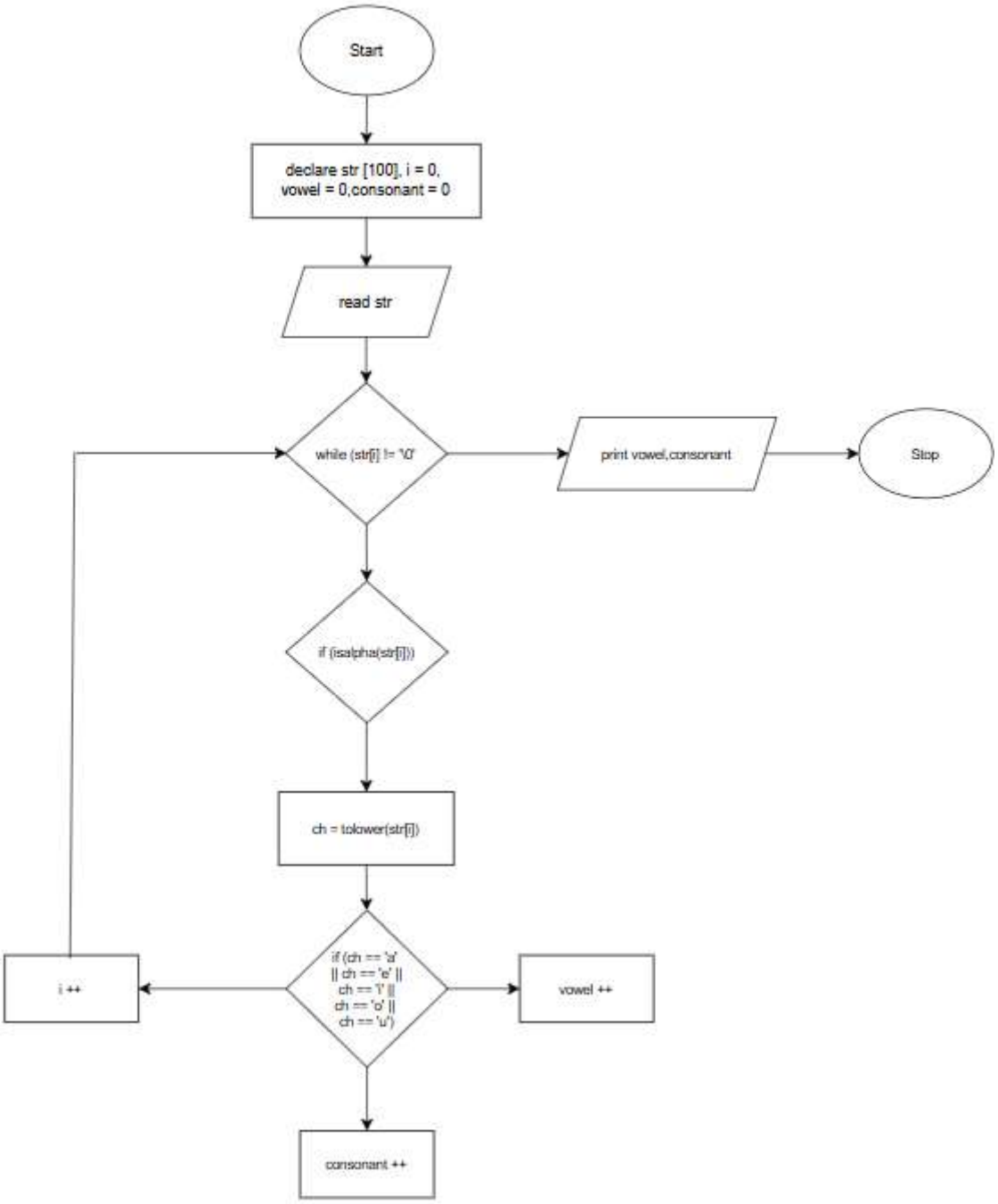
1. Start
2. Take input string from user
3. Initialize `vowel = 0`, `consonant = 0`, index `i = 0`
4. Loop through each character of string
    - If character is a letter:

- If it's a vowel (`a, e, i, o, u`), increment `vowel`
- Else, increment `consonant`

5. Print `vowel` and `consonant` count
6. End

Program

```
#include <stdio.h>
#include <ctype.h>
int main() {
   char str[100];
   int i = 0, vowel = 0, consonant = 0;
   printf("Enter a string: ");
   scanf("%[^\n]s", str); // Reads input until newline (space allowed)
   while (str[i] != '\0') {
      if (isalpha(str[i])) {
         char ch = tolower(str[i]);
         if (ch == 'a' || ch == 'e' || ch == 'i' || ch == 'o' || ch == 'u')
            vowel++;
         else
            consonant++;
      }
      i++;
   }
   printf("Vowels: %d\n", vowel);
   printf("Consonants: %d\n", consonant);
   return 0;
}
```



71.Draw a flow chart to print transpose a matrix.

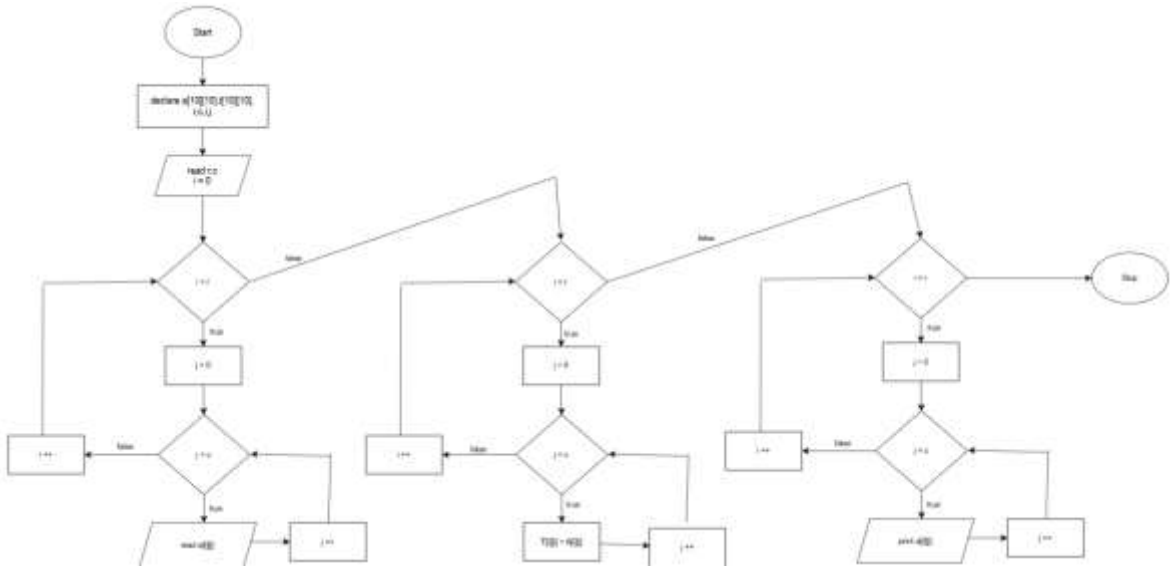
Algorithm:

1. Start
2. Declare two 2D arrays: `A[10][10]`, `T[10][10]`
3. Take input for number of rows `r` and columns `c`
4. Input matrix elements into `A[r][c]`
5. For each element `A[i][j]`, set `T[j][i] = A[i][j]`
6. Print the transposed matrix `T`

7. End

Program:

```
#include <stdio.h>
int main() {
   int A[10][10], T[10][10];
   int r, c, i, j;
   printf("Enter number of rows and columns: ");
   scanf("%d %d", &r, &c);
   printf("Enter elements of matrix:\n");
   for (i = 0; i < r; i++) {
      for (j = 0; j < c; j++) {
         scanf("%d", &A[i][j]);
      }
   }
   for (i = 0; i < r; i++) {
      for (j = 0; j < c; j++) {
         T[j][i] = A[i][j];
      }
   }
   printf("Transposed Matrix:\n");
   for (i = 0; i < c; i++) {
      for (j = 0; j < r; j++) {
         printf("%d ", T[i][j]);
      }
      printf("\n");
   }
   return 0;
}
```



72. Draw a flow chart to print Multiplication of two given matrix .

Algorithm:

1. Start
2. Input matrix sizes: rows1 × cols1, rows2 × cols2
3. Check if `cols1 == rows2`, else print error and stop
4. Input Matrix A (rows1 × cols1)
5. Input Matrix B (rows2 × cols2)
6. Initialize Matrix C[rows1][cols2] = 0
7. Loop i = 0 to rows1 - 1
    - Loop j = 0 to cols2 - 1
        - Loop k = 0 to cols1 - 1
            - C[i][j] += A[i][k] * B[k][j]
8. Print result matrix C
9. End

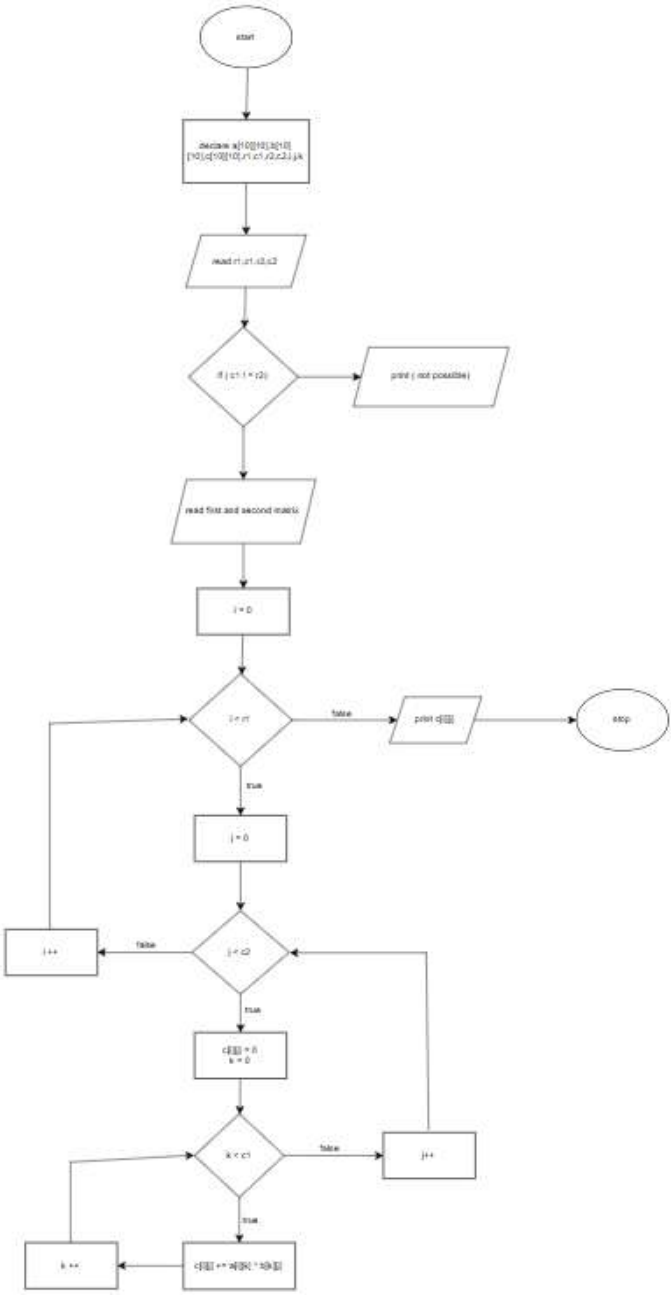
Program

```
#include <stdio.h>
int main() {
   int A[10][10], B[10][10], C[10][10];
   int r1, c1, r2, c2;
   int i, j, k;
   printf("Enter rows and columns for first matrix: ");
```

```
scanf("%d %d", &r1, &c1);
printf("Enter rows and columns for second matrix: ");
scanf("%d %d", &r2, &c2);

if (c1 != r2) {
   printf("Matrix multiplication not possible!\n");
   return 1;
}
printf("Enter elements of first matrix:\n");
for (i = 0; i < r1; i++)
   for (j = 0; j < c1; j++)
      scanf("%d", &A[i][j]);
printf("Enter elements of second matrix:\n");
for (i = 0; i < r2; i++)
   for (j = 0; j < c2; j++)
      scanf("%d", &B[i][j]);
// Multiply
for (i = 0; i < r1; i++) {
   for (j = 0; j < c2; j++) {
      C[i][j] = 0;
      for (k = 0; k < c1; k++) {
         C[i][j] += A[i][k] * B[k][j];
      }
   }
}
printf("Result matrix:\n");
for (i = 0; i < r1; i++) {
   for (j = 0; j < c2; j++)
      printf("%d ", C[i][j]);
   printf("\n");
}
```

```
    return 0;
}
```



73.Draw a flow chart to find hours:minutes:seconds, if user gives seconds

Ex: - I/p = 3600

o/p=01:00:00

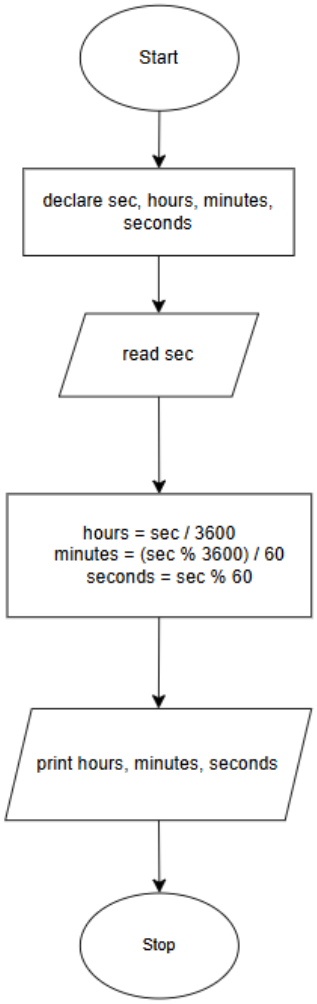
Algorithm:

1. Start
2. Input total seconds `sec`
3. `hours = sec / 3600`

4. `minutes = (sec % 3600) / 60`
5. `seconds = sec % 60`
6. Print `hours:minutes:seconds` in 2-digit format
7. End

Program

```
#include <stdio.h>
int main() {
   int sec, hours, minutes, seconds;
   printf("Enter total seconds: ");
   scanf("%d", &sec);
   hours = sec / 3600;
   minutes = (sec % 3600) / 60;
   seconds = sec % 60;
   printf("%02d:%02d:%02d\n", hours, minutes, seconds);
   return 0;
}
```



74. Draw a flow chart to check whether the given two lines are parallel or not.

Algorithm:

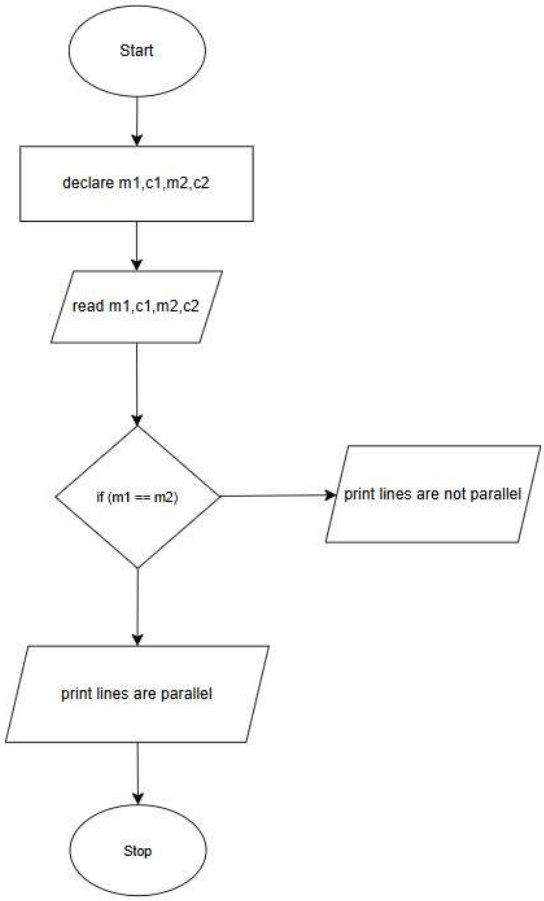
1. Start
2. Input `m1`, `c1` (slope and intercept of line 1)
3. Input `m2`, `c2` (slope and intercept of line 2)
4. If `m1 == m2`
    - Print “Lines are parallel”
      Else
    - Print “Lines are not parallel”
5. End

Program

```
#include <stdio.h>

int main() {
```

```
    float m1, c1, m2, c2;
    printf("Enter slope and intercept of line 1 (m1 c1): ");
    scanf("%f %f", &m1, &c1);
    printf("Enter slope and intercept of line 2 (m2 c2): ");
    scanf("%f %f", &m2, &c2);
    if (m1 == m2)
        printf("Lines are parallel.\n");
    else
        printf("Lines are not parallel.\n");
    return 0;
}
```



75.Draw a flow chart to find sum of lower triangle elements in a matrix?

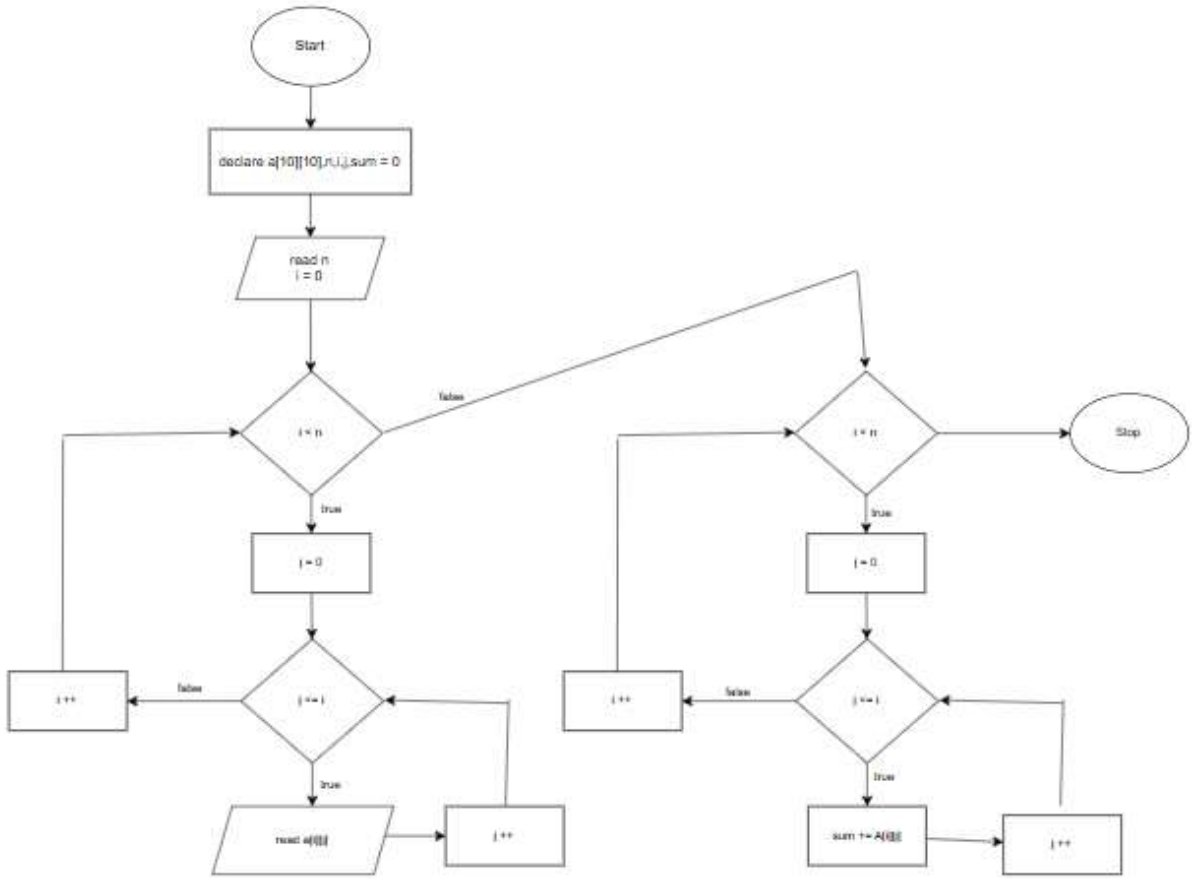
Algorithm:

1. Start
2. Input matrix size `n`

3. Input matrix elements `A[n][n]`
4. Initialize `sum = 0`
5. Loop `i = 0 to n-1`
    - Loop `j = 0 to i` (i.e., j ≤ i)
        - Add `A[i][j]` to `sum`
6. Print `sum`
7. End

Program

```
#include <stdio.h>
int main() {
   int A[10][10], n, i, j, sum = 0;
   printf("Enter size of square matrix (n): ");
   scanf("%d", &n);
   printf("Enter matrix elements:\n");
   for (i = 0; i < n; i++)
      for (j = 0; j < n; j++)
         scanf("%d", &A[i][j]);
   for (i = 0; i < n; i++) {
      for (j = 0; j <= i; j++) {
         sum += A[i][j];
      }
   }
   printf("Sum of lower triangle elements = %d\n", sum);
   return 0;
}
```



76.Draw a flow chart to to find given series

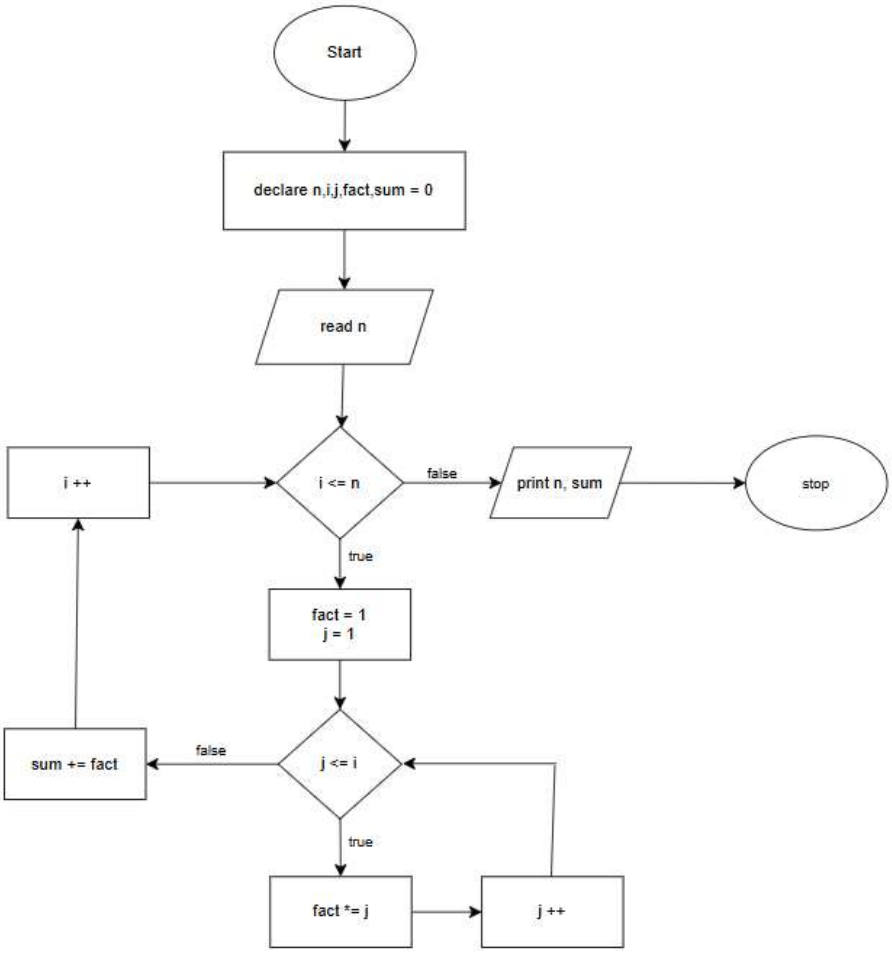
1! +2! +3! +4! +.....+n!

Algorithm:

1. Start
2. Input `n`
3. Initialize `sum = 0`
4. Loop `i = 1 to n`
    - Initialize `fact = 1`
    - Loop `j = 1 to i`: `fact *= j`
    - Add `fact` to `sum`
5. Print `sum`
6. End

Program

```c
#include <stdio.h>
int main() {
    int n, i, j, fact, sum = 0;
    printf("Enter value of n: ");
    scanf("%d", &n);
    for (i = 1; i <= n; i++) {
        fact = 1;
        for (j = 1; j <= i; j++) {
            fact *= j;
        }
        sum += fact;
    }
    printf("Sum of series 1! + 2! + ... + %d! = %d\n", n, sum);
    return 0;
}
```



77.Draw a flow chart to arrange the given string in alphabetical order.

Algorithm:

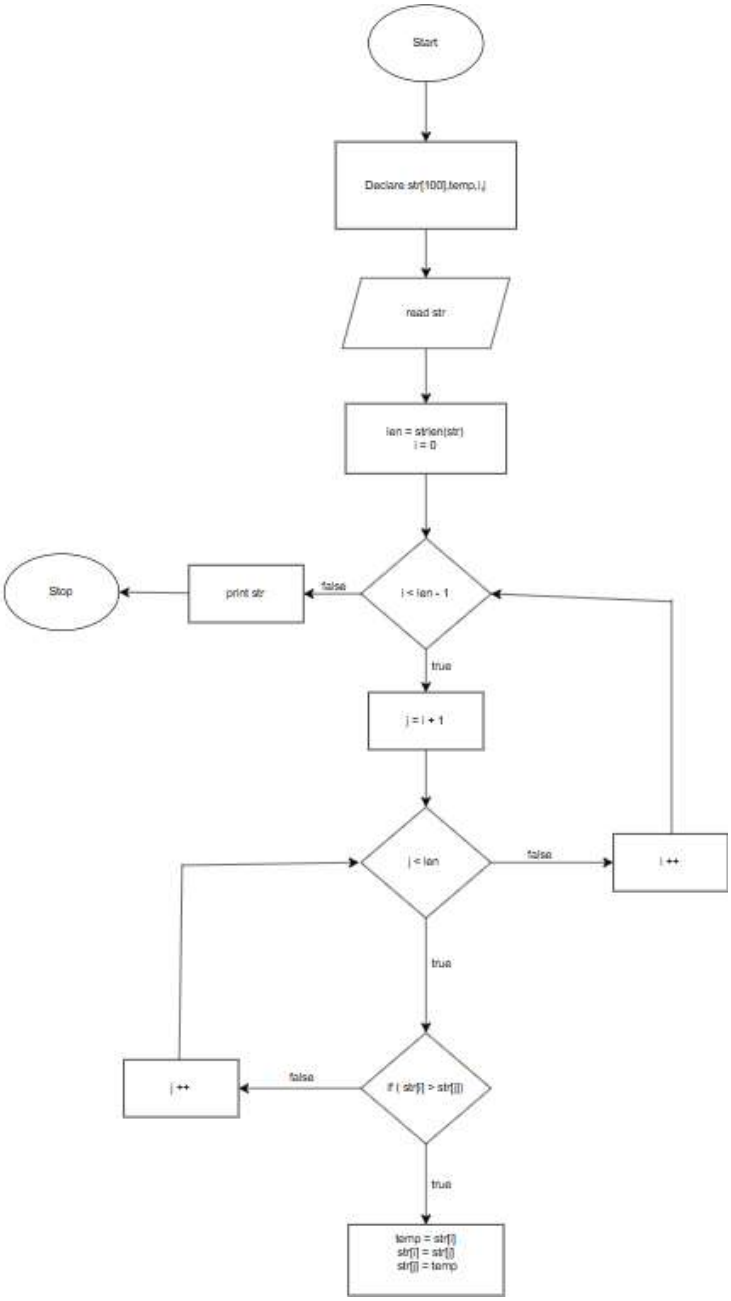
1. Start
2. Input a string `str`
3. Get length of string
4. Use two nested loops:
    - Outer: `i = 0` to `len - 1`
    - Inner: `j = i + 1` to `len`
        - If `str[i] > str[j]`, swap characters
5. Print sorted string
6. End

Program

```
#include <stdio.h>

#include <string.h>
```

```
int main() {
    char str[100], temp;
    int i, j;
    printf("Enter a string: ");
    scanf("%[^\n]s", str);  // Read string with spaces until newline
    int len = strlen(str);
    for (i = 0; i < len - 1; i++) {
        for (j = i + 1; j < len; j++) {
            if (str[i] > str[j]) {
                temp = str[i];
                str[i] = str[j];
                str[j] = temp;
            }
        }
    }
    printf("Sorted string: %s\n", str);
    return 0;
}
```



78.Draw a flow chart to find prime numbers between the given number range taken from user.

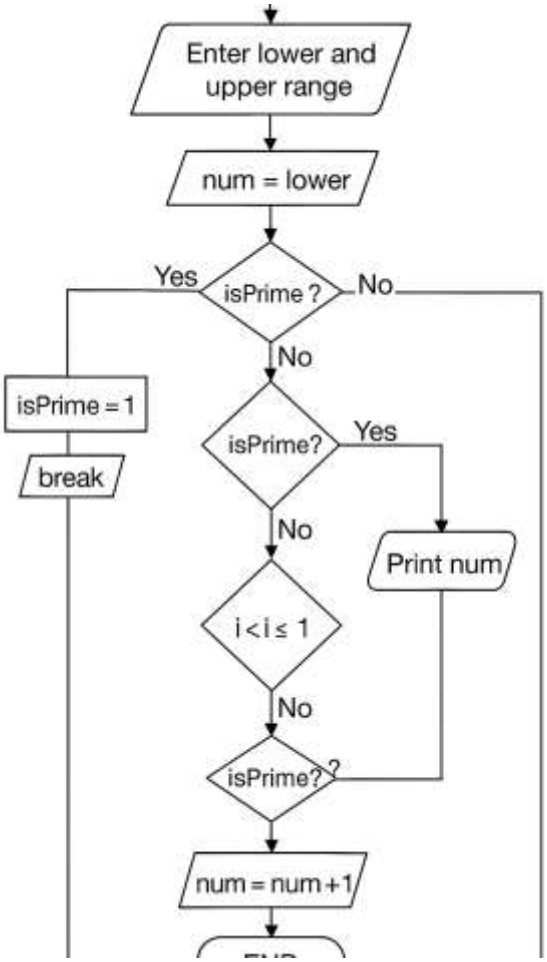
Algorithm:

1. Start
2. Input `lower` and `upper` range
3. Loop `num = lower` to `upper`
    - If `num < 2`: skip
    - Check if `num` is divisible by any number from `2` to `sqrt(num)`
        - If yes → Not prime
        - Else → Print `num`

4. End

Program

```
#include <stdio.h>
int main() {
   int lower, upper, num, i, isPrime;
   printf("Enter lower and upper range: ");
   scanf("%d %d", &lower, &upper);
   printf("Prime numbers between %d and %d are:\n", lower, upper);
   for (num = lower; num <= upper; num++) {
      if (num < 2)
         continue;
      isPrime = 1;
      for (i = 2; i * i <= num; i++) {
         if (num % i == 0) {
            isPrime = 0;
            break;
         }
      }
      if (isPrime)
         printf("%d ", num);
   }
   printf("\n");
   return 0;
}
```



79.Draw a flow chart to find determinant of matrix.

Algorithm:

1. **Start**
2. **Input** 9 elements of the matrix (3 rows × 3 columns)
3. Use the formula:

   det = a11 × (a22×a33 − a23×a32)

   − a12 × (a21×a33 − a23×a31)

   + a13 × (a21×a32 − a22×a31)

4. **Print** the value of `det`

5 . End

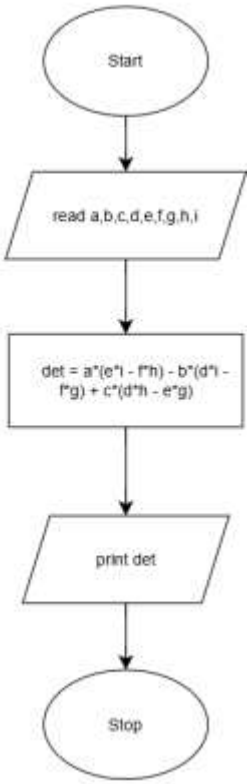
Program

```
#include <stdio.h>

int main() {

   int a, b, c, d, e, f, g, h, i, det;
```

```
    printf("Enter elements of 3x3 matrix (row-wise):\n");
    scanf("%d %d %d", &a, &b, &c);
    scanf("%d %d %d", &d, &e, &f);
    scanf("%d %d %d", &g, &h, &i);
    det = a*(e*i - f*h) - b*(d*i - f*g) + c*(d*h - e*g);
    printf("Determinant of the matrix = %d\n", det);
    return 0;
}
```



80.Draw a flow chart to find the inverse of the matrix.

- Start
- Input 9 elements of the matrix
- Compute the determinant:

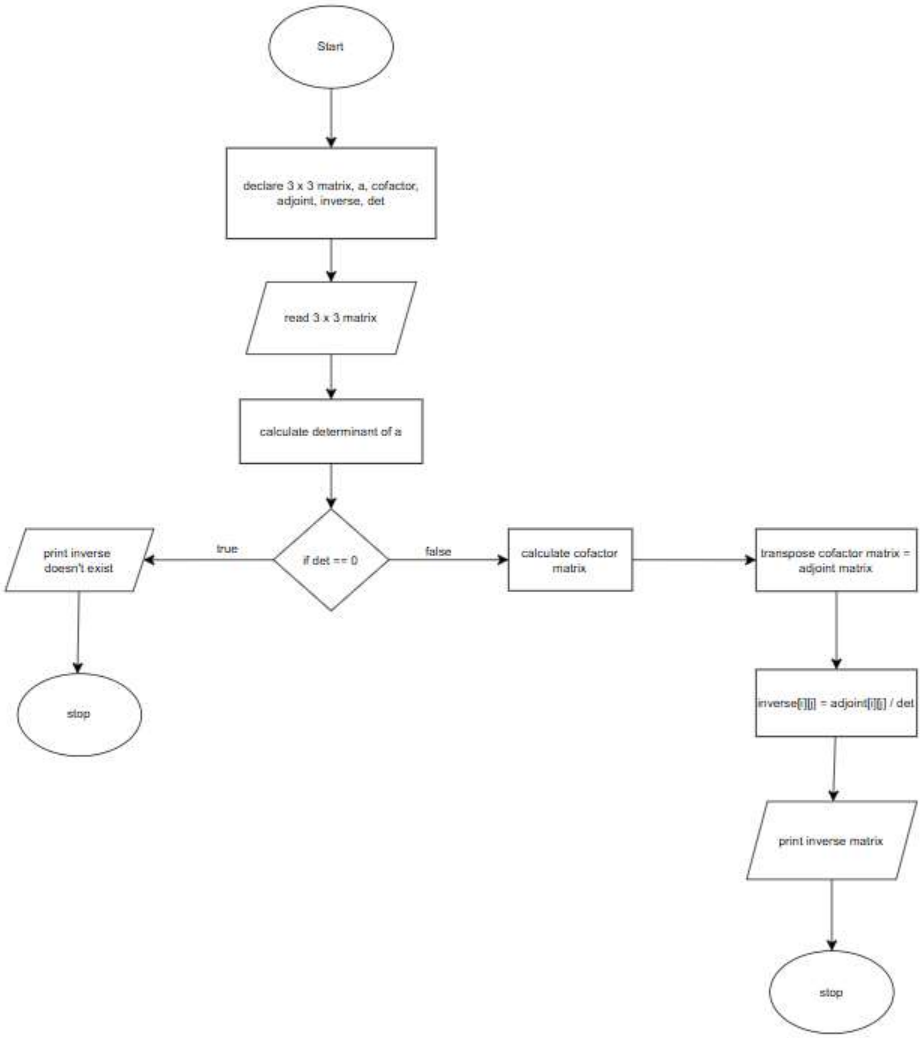
  det = a(ei − fh) − b(di − fg) + c(dh − eg)
- If `det == 0` → Inverse doesn't exist (matrix is singular)
- Compute **cofactor matrix:**

    - Each element is the determinant of the minor, with sign based on position
- **Transpose** the cofactor matrix → this is the **adjoint**
- Divide each element of adjoint by `det` to get the inverse
- Print inverse matrix
- End

Program

```
#include <stdio.h>
int main() {
   float a[3][3], cofactor[3][3], adjoint[3][3], inverse[3][3], det;
   int i, j;
   printf("Enter 3x3 matrix elements:\n");
   for (i = 0; i < 3; i++)
      for (j = 0; j < 3; j++)
         scanf("%f", &a[i][j]);
   // Calculate determinant
   det = a[0][0]*(a[1][1]*a[2][2] - a[1][2]*a[2][1])
      - a[0][1]*(a[1][0]*a[2][2] - a[1][2]*a[2][0])
      + a[0][2]*(a[1][0]*a[2][1] - a[1][1]*a[2][0]);
   if (det == 0) {
      printf("Inverse doesn't exist (Determinant is 0).\n");
      return 1;
   }
   // Cofactor matrix
   cofactor[0][0] =  (a[1][1]*a[2][2] - a[1][2]*a[2][1]);
   cofactor[0][1] = -(a[1][0]*a[2][2] - a[1][2]*a[2][0]);
   cofactor[0][2] =  (a[1][0]*a[2][1] - a[1][1]*a[2][0]);
   cofactor[1][0] = -(a[0][1]*a[2][2] - a[0][2]*a[2][1]);
   cofactor[1][1] =  (a[0][0]*a[2][2] - a[0][2]*a[2][0]);
   cofactor[1][2] = -(a[0][0]*a[2][1] - a[0][1]*a[2][0]);
   cofactor[2][0] =  (a[0][1]*a[1][2] - a[0][2]*a[1][1]);
   cofactor[2][1] = -(a[0][0]*a[1][2] - a[0][2]*a[1][0]);
```

```
    cofactor[2][2] =  (a[0][0]*a[1][1] - a[0][1]*a[1][0]);
    for (i = 0; i < 3; i++)
       for (j = 0; j < 3; j++)
          adjoint[i][j] = cofactor[j][i];
    for (i = 0; i < 3; i++) {
       for (j = 0; j < 3; j++) {
          inverse[i][j] = adjoint[i][j] / det;
       }
    }
    printf("Inverse matrix:\n");
    for (i = 0; i < 3; i++) {
       for (j = 0; j < 3; j++)
          printf("%8.3f ", inverse[i][j]);
       printf("\n");
    }
    return 0;
}
```



81. Draw a flow chart to find leap year count between the given year range taken from user.

Algorithm:

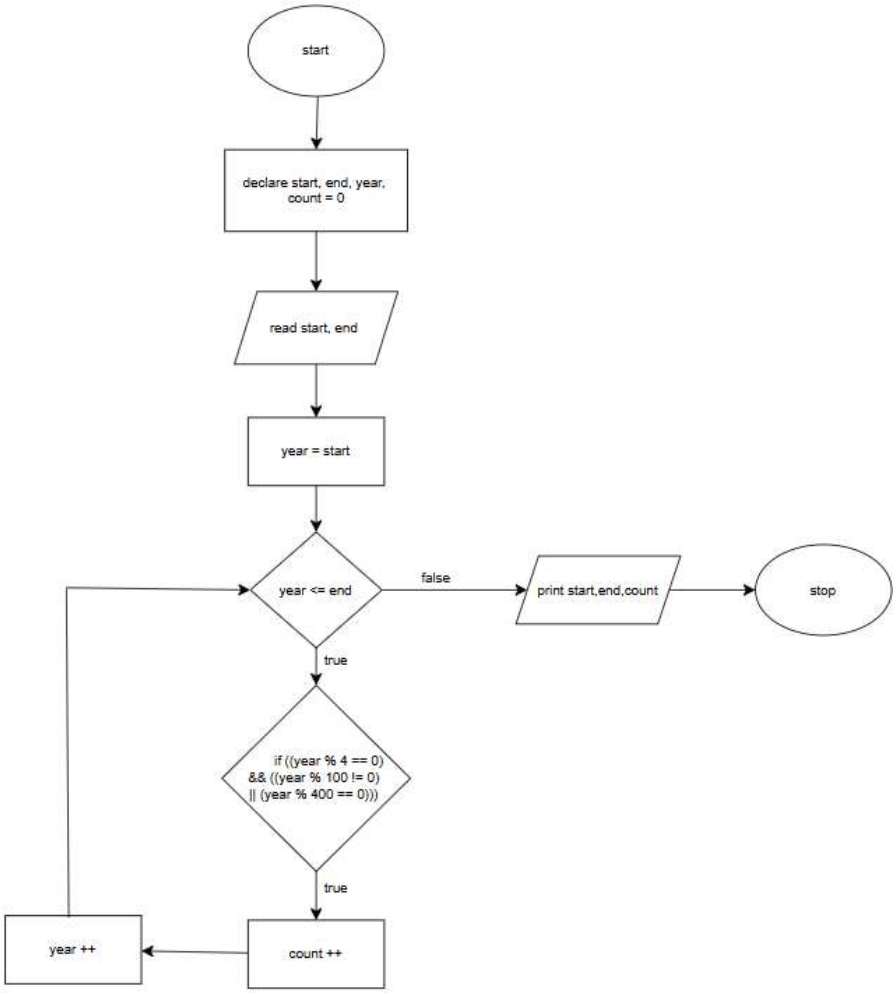
1. Start
2. Input starting year `start`, ending year `end`
3. Initialize `count = 0`
4. Loop from `year = start` to `end`:
    - If `(year % 4 == 0) && ((year % 100 != 0) || (year % 400 == 0))`
      → Increment `count`
5. Print `count`
6. End

Program

```
#include <stdio.h>

int main() {
```

```
    int start, end, year, count = 0;
    printf("Enter starting and ending year: ");
    scanf("%d %d", &start, &end);
    for (year = start; year <= end; year++) {
        if ((year % 4 == 0) && ((year % 100 != 0) || (year % 400 == 0))) {
            count++;
        }
    }
    printf("Number of leap years between %d and %d = %d\n", start, end, count);
    return 0;
}
```



82.Draw a flow chart to find given series

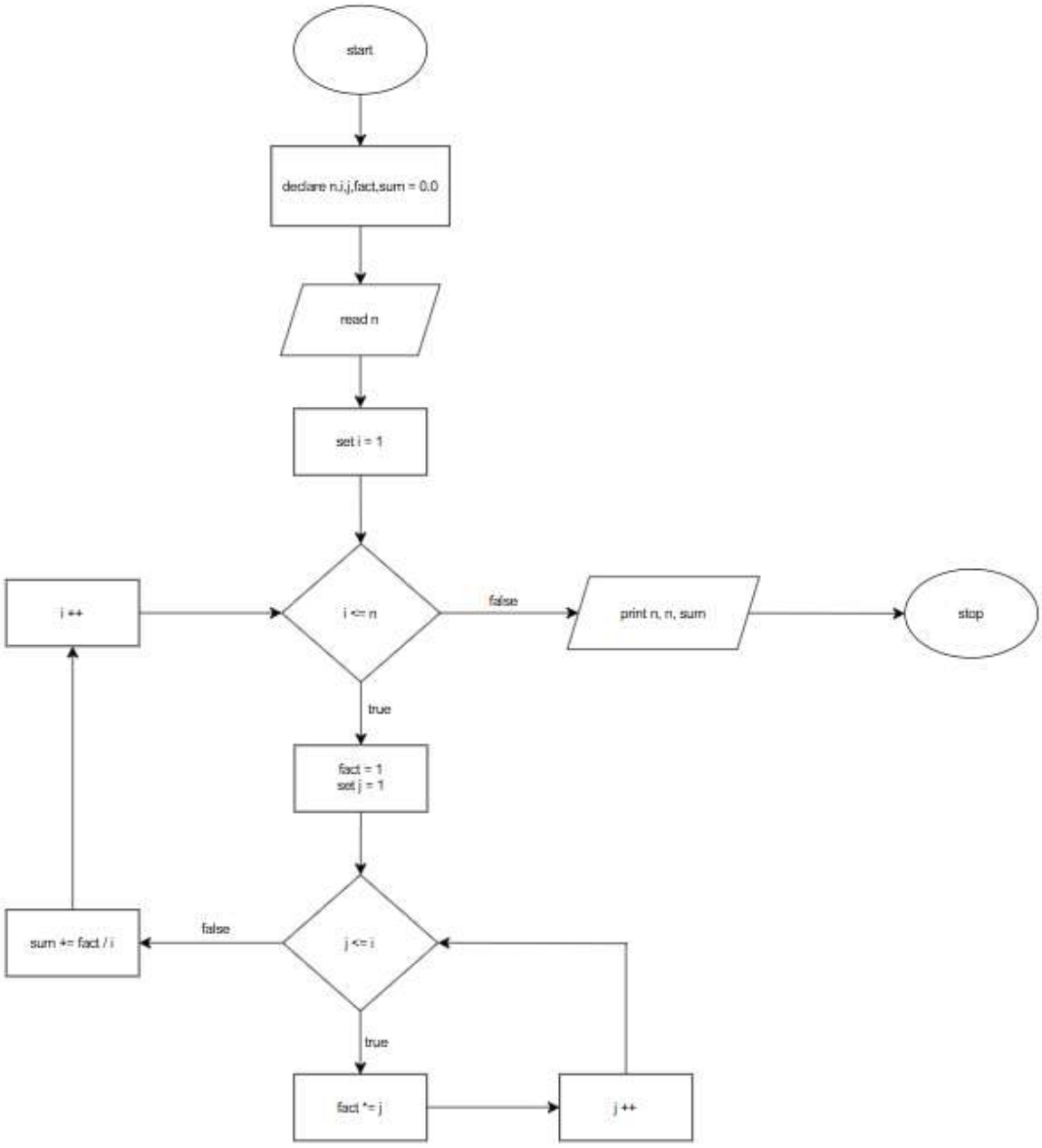
1! /1 + 2! /2 + 3! /3 + ...+ n! /n

Algorithm:

1. Start
2. Input value of `n`
3. Initialize `sum = 0`
4. Loop `i = 1` to `n`
    - Initialize `fact = 1`
    - Loop `j = 1` to `i`: `fact *= j`
    - Add `fact / i` to `sum`
5. Print `sum`
6. End

Program

```
#include <stdio.h>
int main() {
   int n, i, j, fact;
   float sum = 0.0;
   printf("Enter value of n: ");
   scanf("%d", &n);
   for (i = 1; i <= n; i++) {
      fact = 1;
      for (j = 1; j <= i; j++) {
         fact *= j;
      }
      sum += (float)fact / i;
   }
   printf("Sum of series 1!/1 + 2!/2 + ... + %d!/%d = %.2f\n", n, n, sum);
   return 0;
}
```



83. Draw a flow chart to take two complex numbers from user and divide them then print real number and imaginary number

Algorithm:

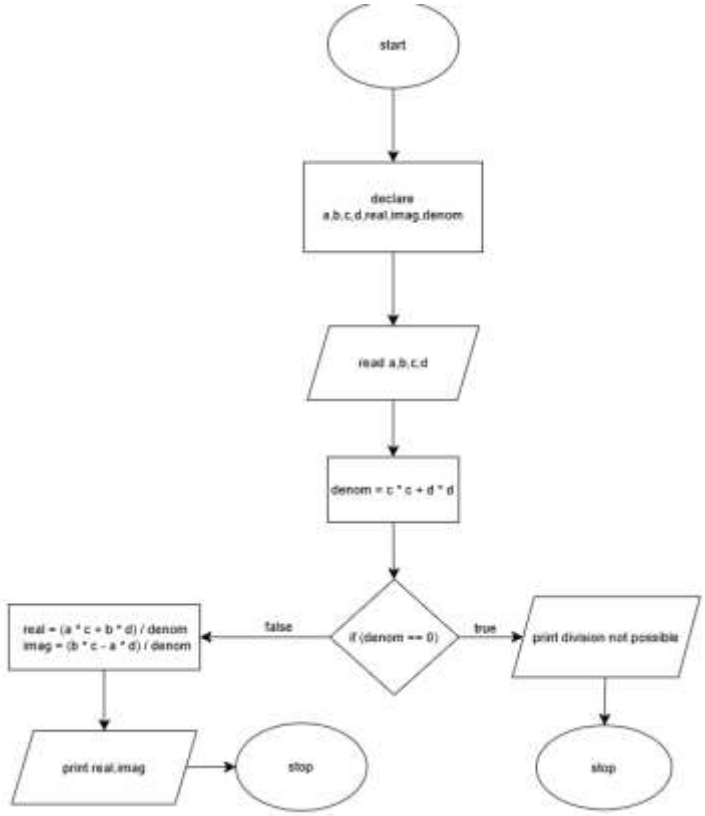
1. Start
2. Input `a`, `b` (real and imag of first number)
3. Input `c`, `d` (real and imag of second number)
4. Compute:
    - denominator = $c^2 + d^2$
    - real part = `(a*c + b*d) / denominator`

- imag part = `(b*c – a*d) / denominator`

5. Print result
6. End

Program

```
#include <stdio.h>
int main() {
   float a, b, c, d;
   float real, imag, denom;
   printf("Enter real and imaginary part of first complex number (a + bi): ");
   scanf("%f %f", &a, &b);
   printf("Enter real and imaginary part of second complex number (c + di): ");
   scanf("%f %f", &c, &d);
   denom = c * c + d * d;
   if (denom == 0) {
      printf("Division not possible. Denominator is zero.\n");
      return 1;
   }
   real = (a * c + b * d) / denom;
   imag = (b * c - a * d) / denom;
   printf("Result of division = %.2f + %.2fi\n", real, imag);
   return 0;
}
```



84.draw a flow chart to print London and American time from given Indian standard time .

Algorithm:

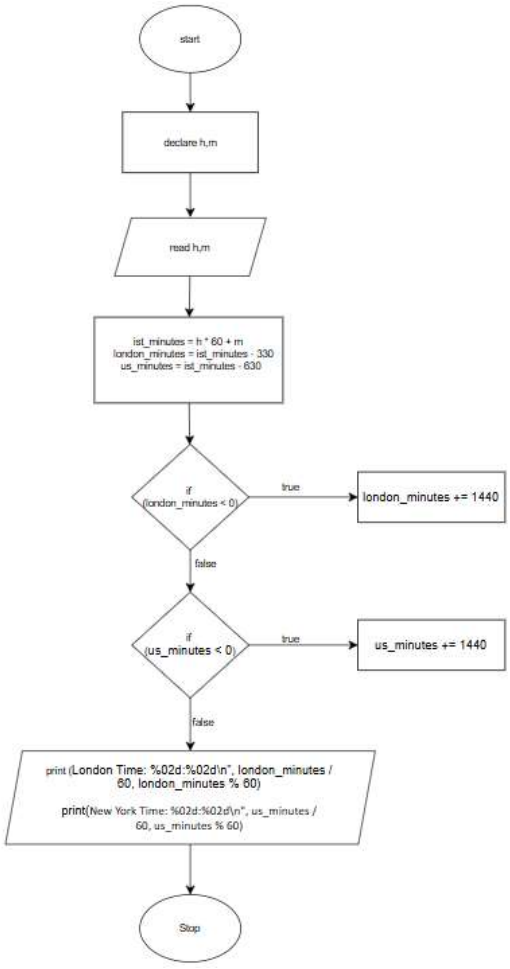
1. Start
2. Input IST time (hours, minutes)
3. Convert IST to total minutes: `total_minutes = h * 60 + m`
4. Compute:
    - London time = `total_minutes - 330`
    - US time = `total_minutes - 630`
5. Convert both to `HH`**:**`MM` format using division/modulo
6. If time < 0, add 1440 (i.e., wrap around 24 hours)
7. Print London and US time
8. End

Program

```
#include <stdio.h>

int main() {
   int h, m;
   printf("Enter IST time (HH MM): ");
```

```
    scanf("%d %d", &h, &m);
    int ist_minutes = h * 60 + m;
    int london_minutes = ist_minutes - 330; // IST to London (UTC+0)
    int us_minutes = ist_minutes - 630;     // IST to New York (UTC-5)
    if (london_minutes < 0)
        london_minutes += 1440;  // 24 * 60
    if (us_minutes < 0)
        us_minutes += 1440;
    printf("London Time: %02d:%02d\n", london_minutes / 60, london_minutes % 60);
    printf("New York Time: %02d:%02d\n", us_minutes / 60, us_minutes % 60);
    return 0;
}
```



85.Draw a flow chart to display LSB from given number.

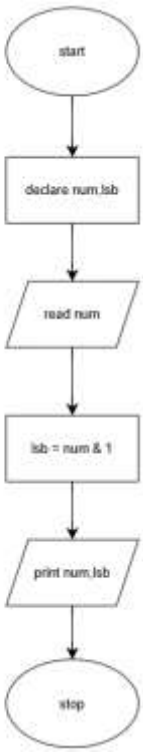
Algorithm:

1. Start
2. Input an integer `num`
3. Compute `lsb = num & 1`
4. Print `lsb`
5. End

Program

```
#include <stdio.h>


int main() {
   int num, lsb;
   printf("Enter a number: ");
   scanf("%d", &num);
   lsb = num & 1;
   printf("LSB of %d = %d\n", num, lsb);
   return 0;
}
```



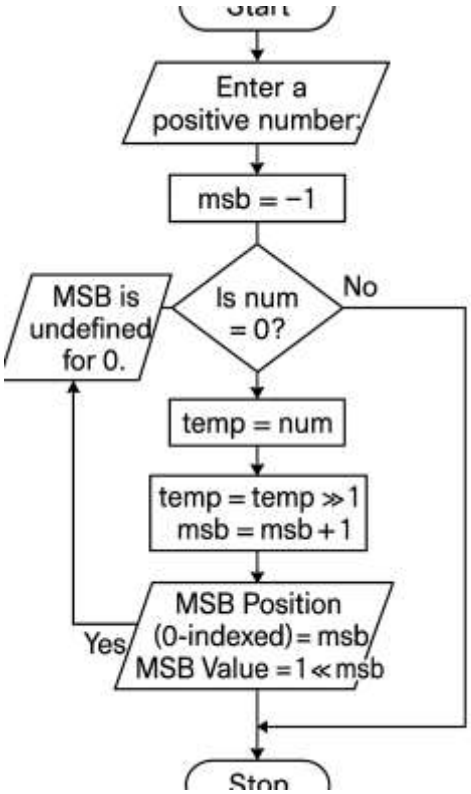
86. Draw a flow chart to display MSB from given number.

Algorithm:

1. Start
2. Input number `num`
3. Set `msb = 0`
4. While `num > 1`
    - `num = num >> 1`
    - Increment `msb`
5. MSB position = `msb`
6. Print position or MSB value = `1 << msb`
7. End

Program

```
#include <stdio.h>
int main() {
   unsigned int num;
   int msb = -1;
   printf("Enter a positive number: ");
   scanf("%u", &num);
   if (num == 0) {
      printf("MSB is undefined for 0.\n");
      return 1;
   }
   unsigned int temp = num;
   while (temp > 0) {
      temp = temp >> 1;
      msb++;
   }
   printf("MSB Position (0-indexed) = %d\n", msb);
   printf("MSB Value = %u\n", 1 << msb);
   return 0;
}
```



87.Draw a flow chart to right shift the bits by n times of given binary number.

Algorithm:

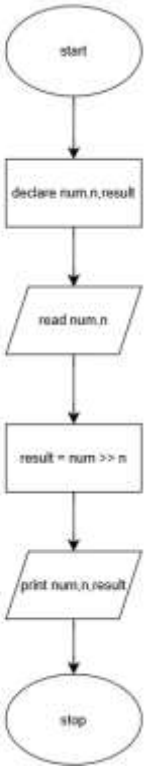
1. Start
2. Input a number `num`
3. Input shift count `n`
4. Compute `result = num >> n`
5. Print `result`
6. End

Program

```
#include <stdio.h>

int main() {
    int num, n, result;
    printf("Enter a number: ");
    scanf("%d", &num);
    printf("Enter how many times to right shift: ");
    scanf("%d", &n);
    result = num >> n;
```

```
    printf("After right shifting %d by %d times: %d\n", num, n, result);
    return 0;
}
```



88.Draw a flow chart to left shift the bits by n times of given binary number.

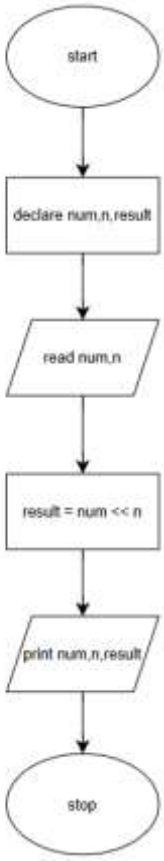
Algorithm:

1. Start
2. Input a number `num`
3. Input shift count `n`
4. Compute `result = num << n`
5. Print `result`
6. End

Program

```
#include <stdio.h>
int main() {
    int num, n, result;
    printf("Enter a number: ");
    scanf("%d", &num);
    printf("Enter how many times to left shift: ");
```

```
    scanf("%d", &n);
    result = num << n;
    printf("After left shifting %d by %d times: %d\n", num, n, result);
    return 0;
}
```



89.Draw a flow chart to toggle the nth bit of given binary number.

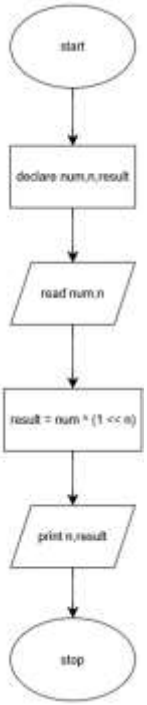
Algorithm:

1. Start
2. Input `num` and bit position `n` (0-indexed)
3. Compute `mask = 1 << n`
4. Toggle bit: `result = num ^ mask`
5. Print result
6. End

Program

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

```
int main() {
    int num, n, result;
    printf("Enter a number: ");
    scanf("%d", &num);
    printf("Enter the bit position to toggle (0-indexed): ");
    scanf("%d", &n);
    result = num ^ (1 << n);
    printf("Number after toggling bit %d: %d\n", n, result);
    return 0;
}
```



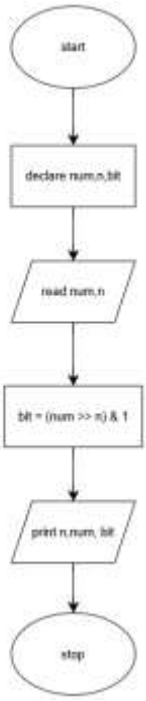
90.Draw a flow chart to display the nth bit of given binary number.

Algorithm:

1. Start
2. Input `num` and bit position `n`
3. Right shift `num` by `n`
4. AND the result with `1`
5. Print the result (0 or 1)
6. End

Program

```
#include <stdio.h>
int main() {
   int num, n, bit;
   printf("Enter a number: ");
   scanf("%d", &num);
   printf("Enter the bit position to display (0-indexed): ");
   scanf("%d", &n);
   bit = (num >> n) & 1;
   printf("The %dth bit of %d is: %d\n", n, num, bit);
   return 0;
}
```



91.Draw a flow chart to display the no.of days between two dates that taken from user.

Algorithm:

1. Start
2. Input two dates: `d1, m1, y1` and `d2, m2, y2`
3. Create function `countDays(d, m, y)` to count total days since year 0:
    - Add days from full years
    - Add days from full months of the current year
    - Add days in current month

4. Subtract the two totals:
   `diff = |countDays(d1, m1, y1) - countDays(d2, m2, y2)|`
5. Print `diff`
6. End

Program

```
#include <stdio.h>
#include <stdlib.h>
int main() {
   int monthDays[12] = {31,28,31,30,31,30,31,31,30,31,30,31};
   int d1, m1, y1, d2, m2, y2;
   printf("Enter first date (DD MM YYYY): ");
   scanf("%d %d %d", &d1, &m1, &y1);
   printf("Enter second date (DD MM YYYY): ");
   scanf("%d %d %d", &d2, &m2, &y2);
   int total1 = y1 * 365 + d1;
   for (int i = 0; i < m1 - 1; i++)
      total1 += monthDays[i];
   total1 += (y1 / 4 - y1 / 100 + y1 / 400);
   if (m1 > 2 && ((y1 % 4 == 0 && y1 % 100 != 0) || (y1 % 400 == 0)))
      total1++;
   int total2 = y2 * 365 + d2;
   for (int i = 0; i < m2 - 1; i++)
      total2 += monthDays[i];
   total2 += (y2 / 4 - y2 / 100 + y2 / 400);
   if (m2 > 2 && ((y2 % 4 == 0 && y2 % 100 != 0) || (y2 % 400 == 0)))
      total2++;

   printf("Number of days between dates: %d\n", abs(total1 - total2));
   return 0;
}
```