**5. Write an algorithm and C program that accepts two integers from the user as input and calculates the sum, difference, product, quotient and remainder applying different arithmetic operators between two integers.**

1. **ALGORITHM**

**STEP 1:** Start

**STEP 2:** Read a, and b

**STEP 3:** Calculate

Sum = a+b

Difference = a-b

Product = a\*b

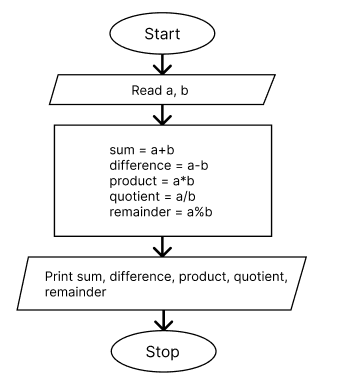
Quotient = a/b

Remainder = a%b

**STEP 4:** Print sum, difference, product, quotient and remainder

**STEP 5:** Stop

1. **FLOWCHART**

****

1. **PROGRAM**

#include<stdio.h>

int main() {

    int a, b;

    printf("Enter two numbers: ");

    scanf("%d %d", &a, &b);

    printf("Sum: %d\n", a + b);

    printf("Difference: %d\n", a - b);

    printf("Product: %d\n", a \* b);

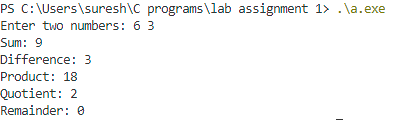
    printf("Quotient: %d\n", a / b);

    printf("Remainder: %d\n", a % b);

    return 0;

}

1. **OUTPUT**

****

**6. Write a C program to convert a given integer (in seconds) to hours, minutes and seconds.**

1. **ALGORITHM**

**STEP 1:** Start

**STEP 2:** Read seconds

**STEP 3:** Calculate hours, minutes, and seconds as

Hours = seconds / 3600

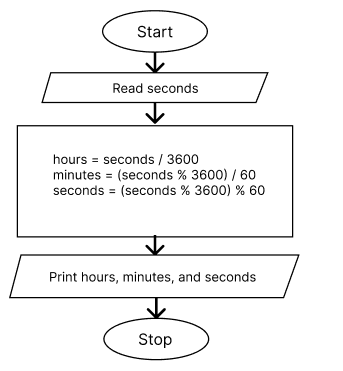
Minutes = (seconds % 3600) / 60

Seconds = (seconds % 3600) % 60

**STEP 4:** Print hours, minutes and seconds

**STEP 5:** Stop

1. **FLOWCHART**

****

1. **PROGRAM**

#include<stdio.h>

int main() {

    int seconds, hours, minutes;

    printf("Enter seconds: ");

    scanf("%d", &seconds);

    hours = seconds / 3600;

    minutes = (seconds % 3600) / 60;

    seconds = (seconds % 3600) % 60;

    printf("Hours: %d\n", hours);

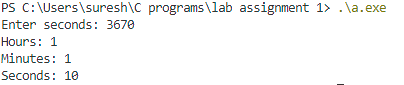
    printf("Minutes: %d\n", minutes);

    printf("Seconds: %d\n", seconds);

    return 0;

}

1. **OUTPUT**

****

**7. Write a C program that accepts principle, rate of interest, time in years and computes the simple interest.**

1. **ALGORITHM**

**STEP 1:** Start

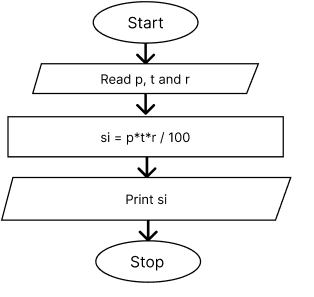
**STEP 2:** Read p, t, and r

**STEP 3:** Calculate si = (p\*t\*r) / 100

**STEP 4:** Print si

**STEP 5:** Stop

1. **FLOWCHART**

****

1. **PROGRAM**

#include <stdio.h>

int main() {

    float p, t, r, si;

    printf("Enter principal, rate and time: ");

    scanf("%f %f %f", &p, &r, &t);

    si = (p \* r \* t) / 100;

    printf("Simple Interest: %.2f", si);

    return 0;

}

1. **OUTPUT**

****

**8. Write algorithm pseudo-code as well as draw flow chart to Compute the roots of the quadratic equation ax2+bx+c =0 for given coefficient input a, b and c. Write C program.**

1. **ALGORITHM**

**STEP 1:** Start

**STEP 2:** Read coefficients a, b and c

**STEP 3:** Calculate d = b\*b- 4ac

**STEP 4:** If d < 0

print the roots are complex numbers and end the program

Else

calculate roots

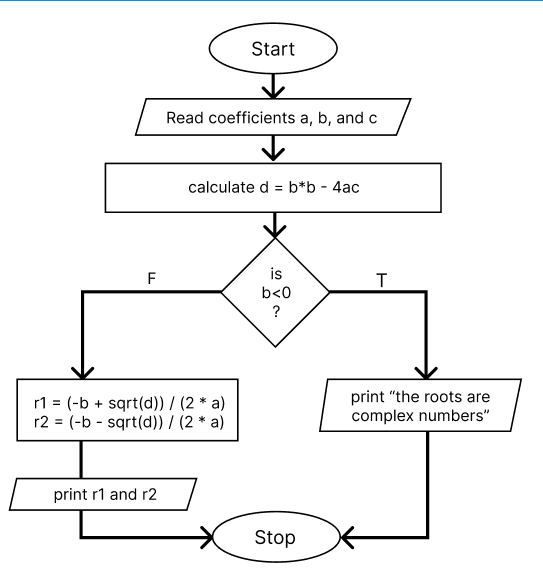
r1 = (-b + sqrt(d)) / 2a and

r­­­2 = (-b – sqrt(d)) / 2a

and print the roots

**STEP 5:** Stop

1. **FLOWCHART**

****

1. **PROGRAM**

#include <stdio.h>

int main()

{

    float a, b, c, d, r1, r2;

    printf("Enter coefficients a, b and c: ");

    scanf("%f %f %f", &a, &b, &c);

    d = b \* b - 4 \* a \* c;

    if (d < 0)

    {

        printf("Roots are complex numbers.\n");

        return 0;

    }

    r1 = (-b + sqrt(d)) / (2 \* a);

    r2 = (-b - sqrt(d)) / (2 \* a);

    printf("Roots are: %.2f and %.2f", r1, r2);

    return 0;

}

1. **OUTPUT**

****

**9. Write a C program to check a given integer is positive even, negative even, positive odd or negative odd.**

1. **ALGORITHM**

**STEP 1:** Start

**STEP 2:** Read n

**STEP 3:** If n > 0

If n%2 == 0

Print the number is positive even

Else

Print the number is positive odd

Else

If n%2 == 0

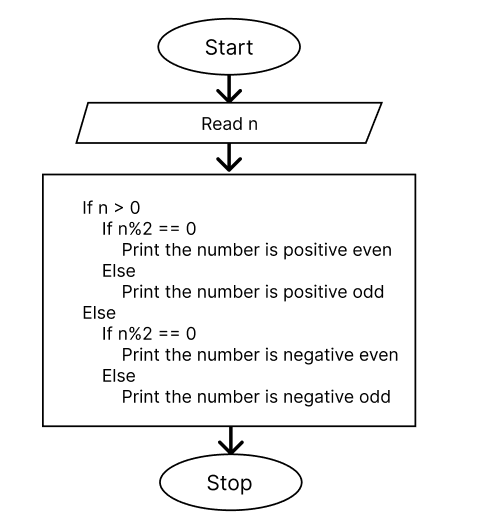
Print the number is negative even

Else

Print the number is negative odd

**STEP 4:** Stop

1. **FLOWCHART**

****

1. **PROGRAM**

#include <stdio.h>

int main()

{

    int n;

    printf("Enter a number: ");

    scanf("%d", &n);

    if (n > 0)

    {

        if (n % 2 == 0)

        {

            printf("%d is positive even\n", n);

        }

        else

        {

            printf("%d is positive odd\n", n);

        }

    }

    else

    {

        if (n % 2 == 0)

        {

            printf("%d is negative even\n", n);

        }

        else

        {

            printf("%d is negative odd\n", n);

        }

    }

    return 0;

}

1. **OUTPUT**

****

**10. Draw a flow chart and write a C program that accepts three integers as input and find the largest of three.**

1. **ALGORITHM**

**STEP 1:** Start

**STEP 2:** Read a, b and c

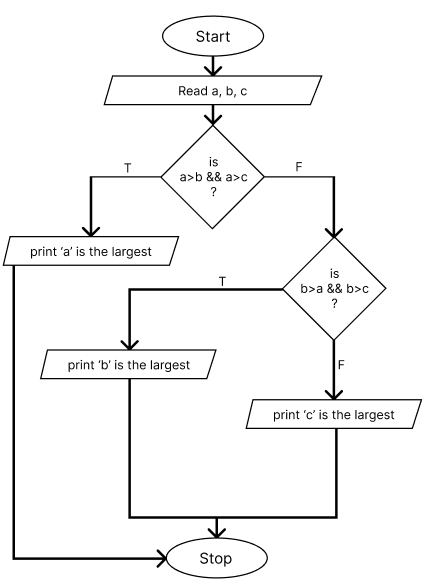
**STEP 3:** If a>b and a<c, print ‘a’ is the largest number

If b>c and b>a, print ‘b’ is the largest number

Else print ‘c’ is the largest number

**STEP 4:** Stop

1. **FLOWCHART**

****

1. **PROGRAM**

#include<stdio.h>

int main() {

    int a, b, c;

    printf("Enter three numbers: ");

    scanf("%d %d %d", &a, &b, &c);

    if (a > b && a > c) {

        printf("%d is the largest number\n", a);

    } else if (b > a && b > c) {

        printf("%d is the largest number\n", b);

    } else {

        printf("%d is the largest number\n", c);

    }

    return 0;

}

1. **OUTPUT**

****

**11. Write a program that takes input of two numbers and an operator in (+,-,\*, /) as input and pass those numbers and an operator to the function. The function should calculate the result of two numbers as indicated by operator and return the result. Display the result of computation in your program.**

1. **ALGORITHM**

**STEP 1:** Start

**STEP 2:** Read two integers a, b

**STEP 3:** Read a choice

**STEP 4:** if choice is ‘+’ return a+b

if choice is ‘-’ return a-b

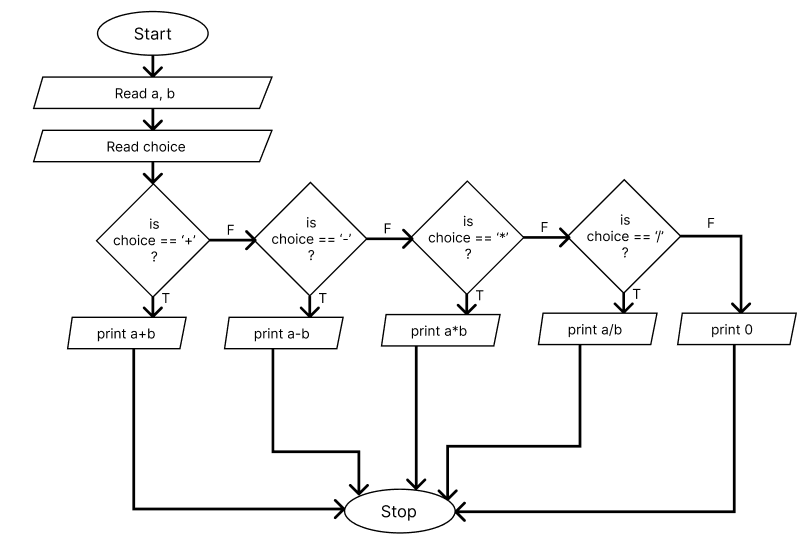
if choice is ‘\*’ return a\*b

if choice is ‘/’ return a/b

else return 0

**STEP 5:** Stop

1. **FLOWCHART**

****

1. **PROGRAM**

#include <stdio.h>

int calculate(int a, int b, char choice) {

    int result;

    switch (choice) {

    case '+':

        result = a + b;

        break;

    case '-':

        result = a - b;

        break;

    case '\*':

        result = a \* b;

        break;

    case '/':

        result = a / b;

        break;

    default:

        result = 0;

    }

    return result;

}

int main() {

    int a, b, result;

    char choice;

    printf("Enter two numbers: ");

    scanf("%d %d", &a, &b);

    printf("Enter your choice +, - \* or /: ");

    scanf(" %c", &choice);

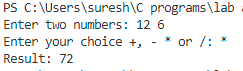
    result = calculate(a, b, choice);

    printf("Result: %d\n", result);

    return 0;

}

1. **OUTPUT**

****

**12. Write a program to determine whether a given number is palindrome or not.**

**a) ALGORITHM**

**STEP 1:** Start

**STEP 2:** Read n

**STEP 3:** temp = n

**STEP 4:** if temp != 0 go to step 5

Else go to step 6

**STEP 5:** rem = temp % 10

rev = rev \* 10 + rem

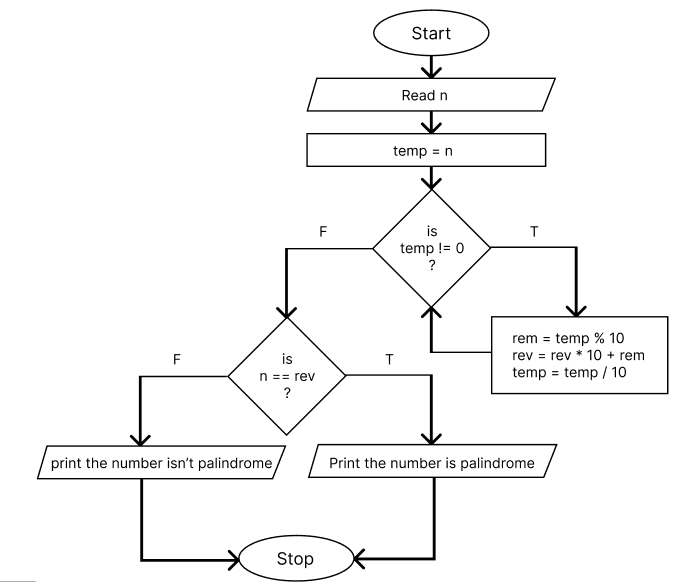
temp = temp / 10

**STEP 6:** if n == rev print “the number is palindrome”

else print “the number is not palindrome”

**STEP 7:** Stop

**b) FLOWCHART**

****

**c) PROGRAM**

#include<stdio.h>

int main() {

    int num, input, rem, rev = 0;

    printf("Enter a number: ");

    scanf("%d", &input);

    num = input;

    while(num != 0) {

        rem = num % 10;

        rev = rev \* 10 + rem;

        num /= 10;

    }

    if(input == rev) {

        printf("%d is a palindrome number.", input);

    } else {

        printf("%d is not a palindrome number.", input);

    }

    return 0;

}

1. **OUTPUT**

****

**13. Write a program to determine whether a given number is Armstrong number or not.**

1. **ALGORITHM**

**STEP 1:** Start

**STEP 2:** Declare num, temp\_num, input, result = 0, rem, num\_of\_digit = 0

**STEP 3:** Read input

**STEP 4:** num = input

temp\_num = input

**STEP 5:** if temp\_num != 0 go to step 6

else go to step 7

**STEP 6:** num\_of\_digit++

temp\_num = temp\_num / 10

go to step 5

**STEP 7:** if num != 0 go to step 8

else go to step 12

**STEP 8:** rem = num % 10

int power = 1

int i = 1

**STEP 9:** if i < num\_of\_digit go to step 10

else go to step 11

**STEP 10:** power = power \* rem

i++

go to step 9

**STEP 11:** result = result + power

num = num / 10

**STEP 12:** if input == result

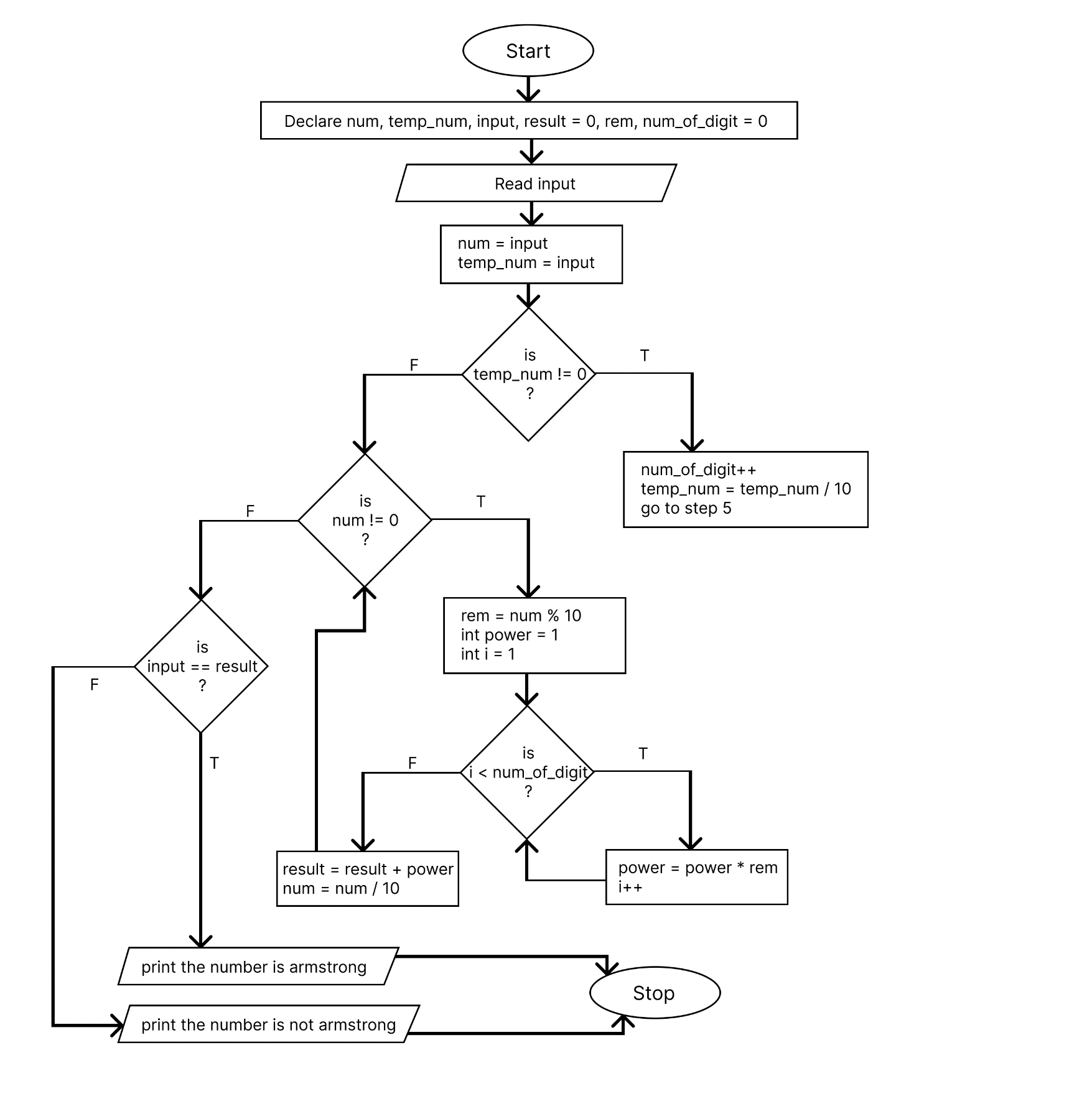
print the number is armstrong

else

print the number is not Armstrong

**STEP 13:** Stop

1. **FLOWCHART**

****

1. **PROGRAM**

#include <stdio.h>

int main()

{

    int num, temp\_num, input, result = 0, rem, num\_of\_digit = 0;

    printf("Enter a number: ");

    scanf("%d", &input);

    num = input;

    temp\_num = input;

    while (temp\_num != 0)

    {

        num\_of\_digit++;

        temp\_num /= 10;

    }

    while (num != 0)

    {

        rem = num % 10;

        int power = 1;

        for (int i = 0; i < num\_of\_digit; i++)

        {

            power \*= rem;

        }

        result += power;

        num /= 10;

    }

    if (input == result)

        printf("%d is an Armstrong number.", input);

    else

        printf("%d is not an Armstrong number.", input);

    return 0;

}

1. **OUTPUT**

****

**14. Write a C program to find the eligibility of admission for a professional course based on the following criteria:**

**Marks in Maths >=65**

**Marks in Phy >=55**

**Marks in Chem>=50**

**Total in all three subject >=180 or Total in Math and physics Subjects >=130**

1. **ALGORITHM**

**STEP 1:** Start

**STEP 2:** Read marks of math, physics and chemistry

**STEP 3:** Find total marks total = math+physics+chemistry

**STEP 4:** if phy >= 55 && chem >= 50 && math >= 65 && total >= 180 && phy + math >= 130

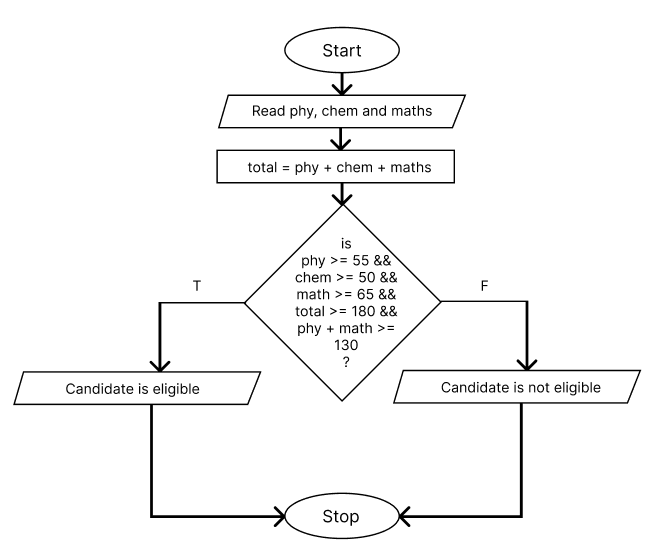
Print the candidate is eligible for admission

else

print the candidate is not eligible for admission

**STEP 5:** Stop

1. **FLOWCHART**

****

1. **PROGRAM**

#include<stdio.h>

int main() {

    float phy, chem, math, total;

    printf("Enter the marks of Physics, Chemistry and Mathematics: ");

    scanf("%f %f %f", &phy, &chem, &math);

    total = phy + chem + math;

    if (

phy >= 55 &&

chem >= 50 &&

math >= 65 &&

total >= 180 &&

phy + math >= 130

)

{

        printf("The candidate is eligible for admission.");

    } else {

        printf("The candidate is not eligible for admission.");

    }

    return 0;

}

1. **OUTPUT**

****

**15. Write a C program to find the sum of first 100 natural numbers using loop.**

1. **ALGORITHM**

**STEP 1:** Start

**STEP 2:** Initialize i = 1, sum = 0

**STEP 3:** if i <= 100 go to step 4

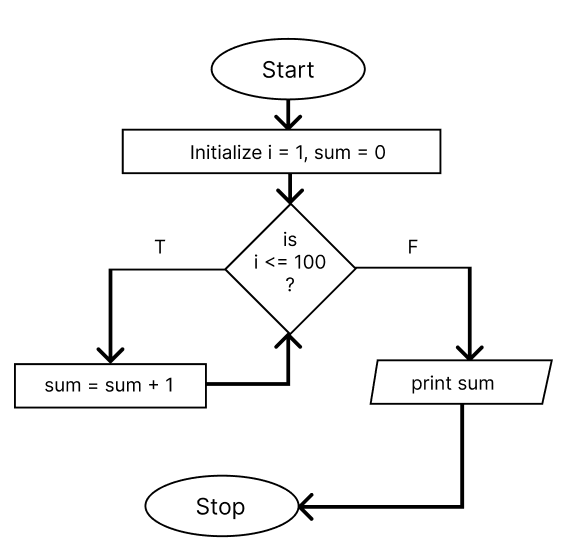
else go to step 5

**STEP 4:** sum = sum + 1

**STEP 5:** Print sum

**STEP 6:** Stop

1. **FLOWCHART**

****

1. **PROGRAM**

#include<stdio.h>

int main() {

    int sum = 0;

    for (int i = 1; i <= 100; i++) {

        sum += i;

    }

    printf("The sum of first 100 natural numbers is %d.", sum);

    return 0;

}

1. **OUTPUT**

****

**16. Write a program in C to display the multiplication table of a given integer.**

1. **ALGORITHM**

**STEP 1:** Start

**STEP 2:** Read an integer n

**STEP 3:** Initialize i = 1

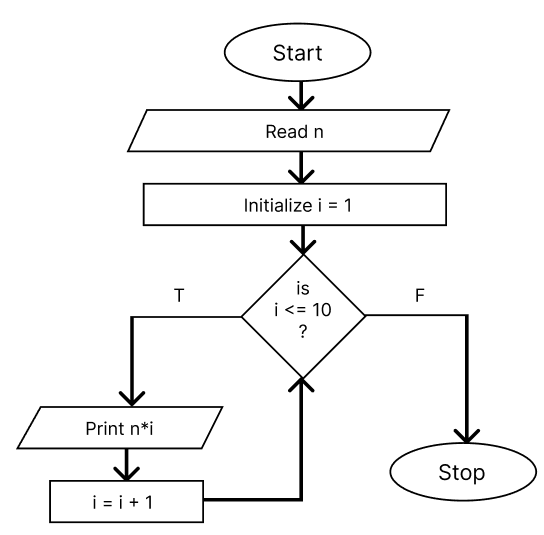
**STEP 4:** if i<=10 go to step 5

else go to step 6

**STEP 5:** print n \* i

**STEP 6:** Stop

1. **FLOWCHART**

****

1. **PROGRAM**

#include<stdio.h>

int main() {

    int n;

    printf("Enter the number: ");

    scanf("%d", &n);

    for (int i = 1; i <= 10; i++) {

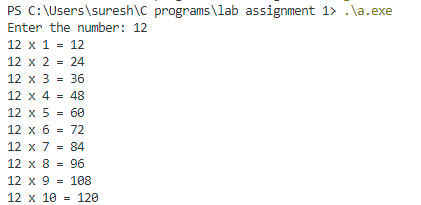
        printf("%d x %d = %d\n", n, i, n \* i);

    }

    return 0;

}

1. **OUTPUT**

****

**17. Write an algorithm/ program to print the prime numbers up to 100.**

1. **ALGORITHM**

**STEP 1:** Start

**STEP 2:** Initialize i = 2

**STEP 3:** If i<=100 go to step 4

Else go to step 5

**STEP 4:** Initialize flag = 0, j = 2

**STEP 5:** If j<=i/2 go to step 6

Else go to step 7

**STEP 6:** if i % j == 0

Assign 1 to flag variable

j = j+1

Go to step 7

**STEP 7:** if flag == 0

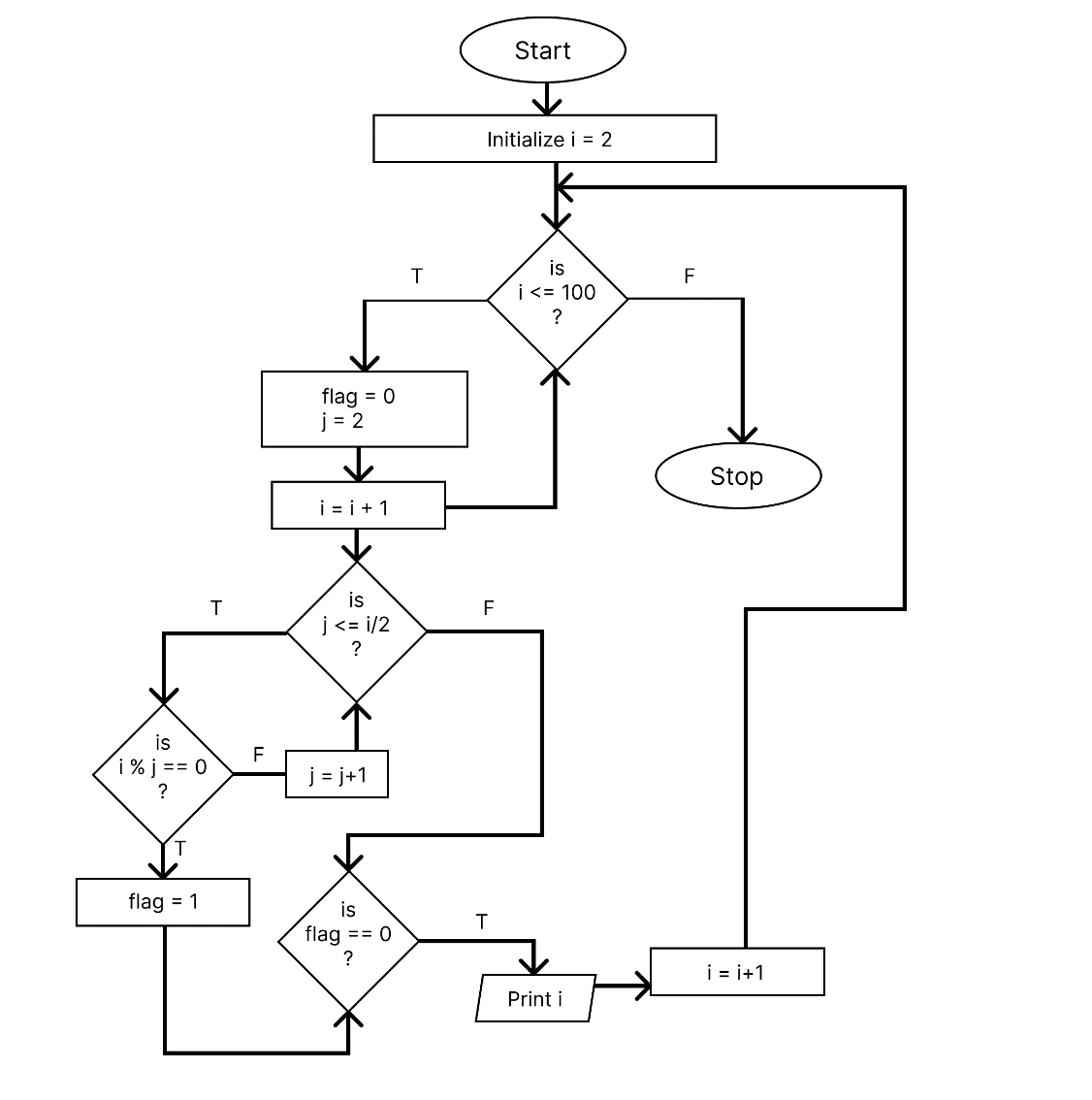
print i

i = i+1

go to step 3

**STEP 8:** Stop

1. **FLOWCHART**

****

1. **PROGRAM**

#include<stdio.h>

int main() {

    int n, flag;

    printf("Prime numbers between 1 to 100 are:\n");

    for (int i = 2; i <= 100; i++) {

        flag = 0;

        for (int j = 2; j <= i / 2; j++) {

            if (i % j == 0) {

                flag = 1;

                break;

            }

        }

        if (flag == 0) {

            printf("%d, ", i);

        }

    }

    return 0;

}

1. **OUTPUT**

****

**18. Write algorithm and program to compute the followings using for, do while and while loop separately.**

* 1. **factorial of an integer N**
  2. **computation of ab( a raised to power b)**

1. **factorial of an integer using for loop**
   * 1. **ALGORITHM**

**STEP 1:** Start

**STEP 2:** Read n, set fact = 1, i = 1

**STEP 3:** if i<=n go to step 4 else go to step 5

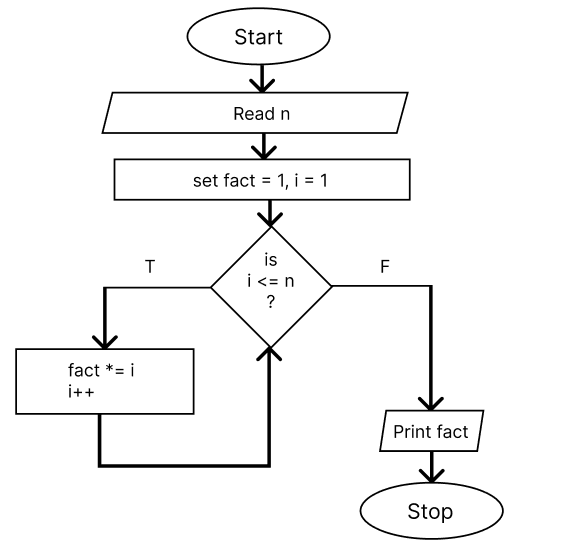
**STEP 4:** fact = fact \* i

i++

**STEP 5:** Print fact

**STEP 6:** Stop

* + 1. **FLOWCHART**

****

* + 1. **PROGRAM**

#include<stdio.h>

int main() {

    int n, fact = 1;

    printf("Enter a number: ");

    scanf("%d", &n);

    for (int i = 1; i <= n; i++) {

        fact \*= i;

    }

    printf("Factorial is %d", fact);

    return 0;

}

* + 1. **OUTPUT**

****

1. **factorial of an integer using while loop**
2. **ALGORITHM**

**STEP 1:** Start

**STEP 2:** Read n, set fact = 1

**STEP 3:** if n > 0 go to step 4 else go to step 5

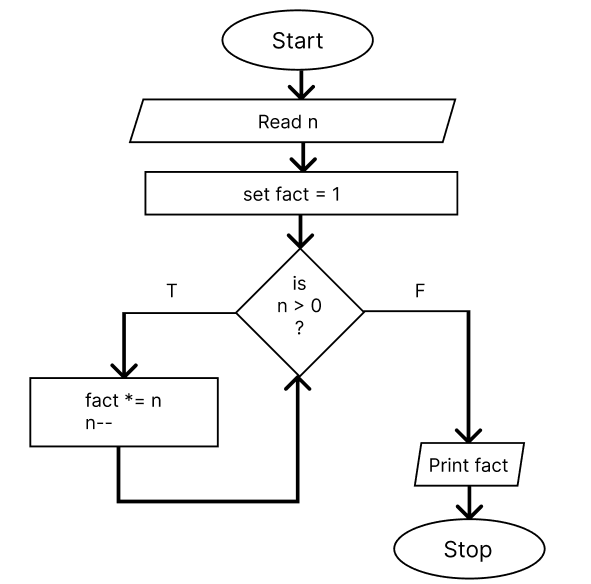
**STEP 4:** fact = fact \* n

n = n - 1

**STEP 5:** Print fact

**STEP 6:** Stop

1. **FLOWCHART**

****

1. **PROGRAM**

#include<stdio.h>

int main() {

    int n, fact = 1;

    printf("Enter a number: ");

    scanf("%d", &n);

    while(n > 0) {

        fact \*= n;

        n--;

    }

    printf("Factorial is %d", fact);

    return 0;

}

1. **OUTPUT**



1. **factorial of an integer using do while loop**
2. **ALGORITHM**

**STEP 1:** Start

**STEP 2:** Read n, set fact = 1

**STEP 3:** fact = fact \* n

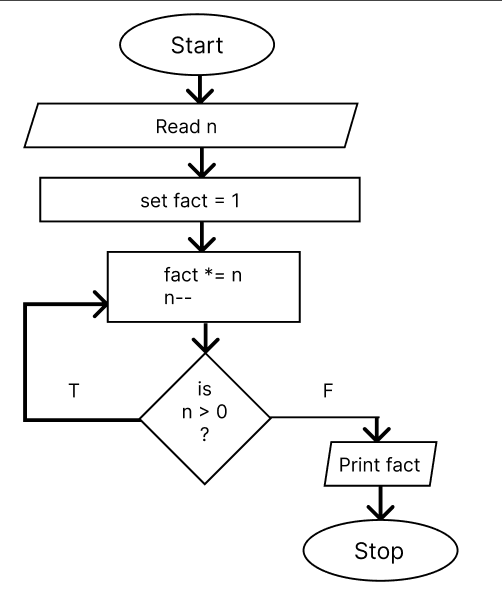
n = n - 1

**STEP 4:** if n > 0 go to step 3 else go to step 5

**STEP 5:** Print fact

**STEP 6:** Stop

1. **FLOWCHART**

****

1. **PROGRAM**

#include<stdio.h>

int main() {

    int n, fact = 1;

    printf("Enter a number: ");

    scanf("%d", &n);

    do {

        fact \*= n;

        n--;

    } while (n > 0);

    printf("Factorial is %d", fact);

    return 0;

}

1. **OUTPUT**



1. **computation of ab( a raised to power b)**
2. **Computation of ab using for loop**
3. **ALGORITHM**

**STEP 1:** Start

**STEP 2:** Declare power, base, result = 1

**STEP 3:** Read power and base

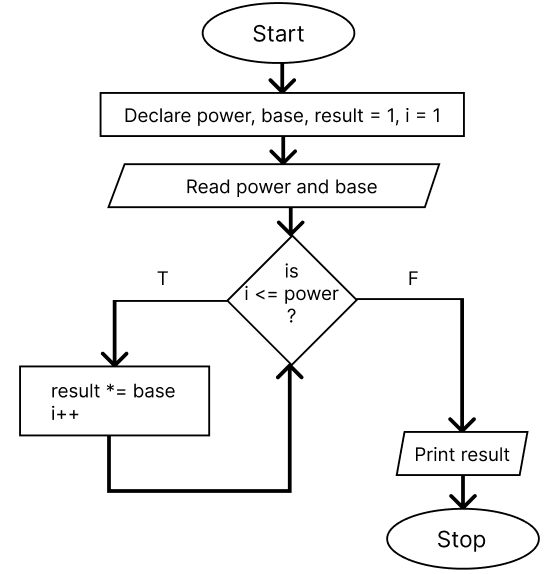
**STEP 4:** if i <= power go to step 5 else go to step 6

**STEP 5:** result = result \* base

**STEP 6:** Print result

**STEP 7:** Stop

1. **FLOWCHART**

****

1. **PROGRAM**

#include<stdio.h>

int main() {

    int power, base, result = 1;

    printf("Enter a number and its power: ");

    scanf("%d %d", &base, &power);

    for (int i = 1; i <= power; i++) {

        result \*= base;

    }

    printf("%d ^ %d is %d", base, power, result);

    return 0;

}

1. **OUTPUT**

****

1. **Computation of ab using while loop**
2. **ALGORITHM**

**STEP 1:** Start

**STEP 2:** Declare power, base, temp, result = 1

**STEP 3:** Read power and base

**STEP 4:** temp = power

**STEP 5:** if temp > 0 go to step 6 else go to step 7

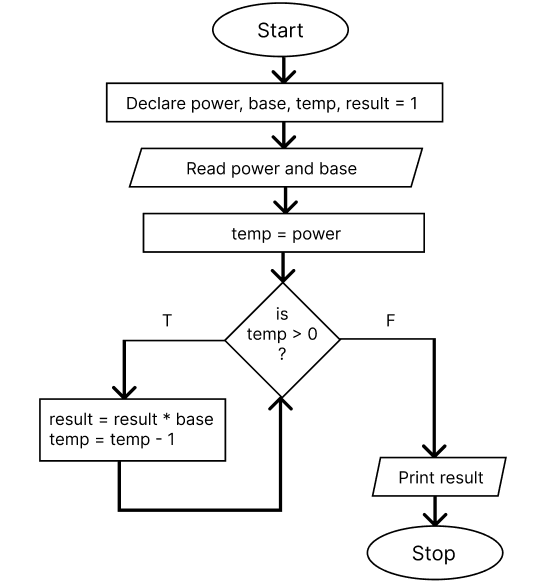
**STEP 6:** result = result \* base

temp = temp - 1

**STEP 7:** Print result

**STEP 8:** Stop

1. **FLOWCHART**

****

1. **PROGRAM**

#include<stdio.h>

int main() {

    int base, temp\_exponent, exponent, result = 1;

    printf("Enter base: ");

    scanf("%d", &base);

    printf("Enter exponent: ");

    scanf("%d", &exponent);

    temp\_exponent = exponent;

    while (temp\_exponent > 0) {

        result \*= base;

        temp\_exponent--;

    }

    printf("%d^%d = %d", base, exponent, result);

    return 0;

}

1. **OUTPUT**



1. **Computation of ab using do while loop**
2. **ALGORITHM**

**STEP 1:** Start

**STEP 2:** Declare power, base, temp, result = 1

**STEP 3:** Read power and base

**STEP 4:** temp = power

**STEP 5:** result = result \* base

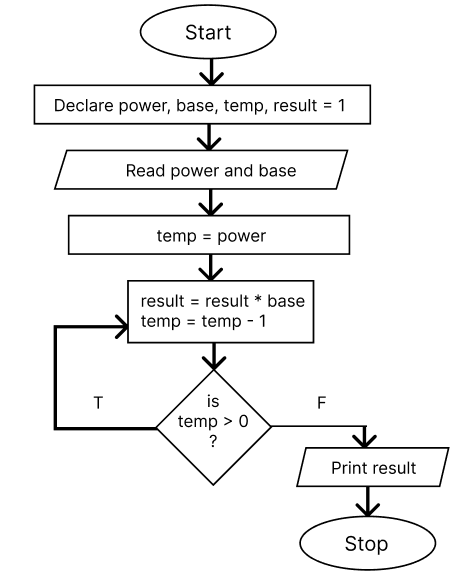
temp = temp - 1

**STEP 6:** if temp > 0 go to step 5 else go to step 7

**STEP 7:** Print result

**STEP 8:** Stop

1. **FLOWCHRT**

****

1. **PROGRAM**

#include<stdio.h>

int main() {

    int base, temp\_exponent, exponent, result = 1;

    printf("Enter a number and its power: ");

    scanf("%d %d", &base, &exponent);

    temp\_exponent = exponent;

    do {

        result \*= base;

        temp\_exponent--;

    } while (temp\_exponent > 0);

    printf("%d^%d = %d", base, exponent, result);

    return 0;

}

1. **OUTPUT**



**19. Write a program in C to make such a pattern of astrisk(\*) below using loop.**

**\***

**\* \***

**\* \* \***

**\* \* \* \* up to n lines where n is an integers**

1. **ALGORITHM**

**STEP 1:** Start

**STEP 2:** Declare n, i = 1, j = 1

**STEP 3:** Read n

**STEP 4:** If i <= n

go to step 5

else go to step 8

**STEP 5:** If j <= i

print ‘\*’

j++

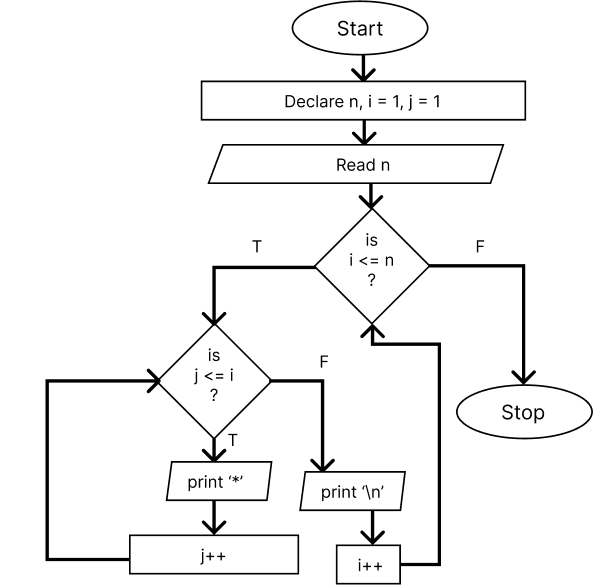
else go to step 6

**STEP 6:** Print “\n”

**STEP 7:** i++ and go to step 4

**STEP 8:** Stop

1. **FLOWCHART**

****

1. **PROGRAM**

#include<stdio.h>

int main() {

    int n, i, j;

    printf("Enter a number: ");

    scanf("%d", &n);

    for (i = 1; i <= n; i++) {

        for (j = 1; j <= i; j++) {

            printf("\*");

        }

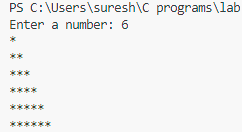
        printf("\n");

    }

    return 0;

}

**d) OUTPUT**

****

**20. Write a program using loop to print the following Floyd’s triangle as given below when input is n.**

**1**

**2 3**

**4 5 6**

**7 8 9 10**

**11 12 13 14 up to n rows**

1. **ALGORITHM**

**STEP 1:** Start

**STEP 2:** Declare n, i = 1, j = 1, count = 1

**STEP 3:** Read n

**STEP 4:** If i <= n

go to step 5

else go to step 8

**STEP 5:** If j <= i

print count

count++

j++

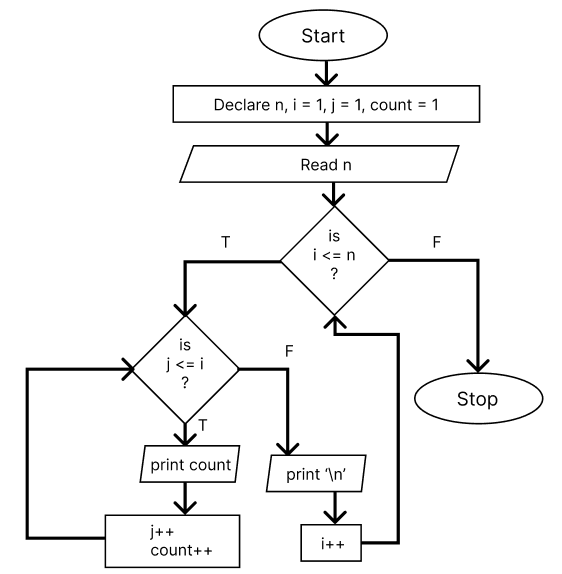
else go to step 6

**STEP 6:** Print “\n”

**STEP 7:** i++ and go to step 4

**STEP 8:** Stop

1. **FLOWCHART**

****

1. **PROGRAM**

#include<stdio.h>

int main() {

    int n, i, j, count = 1;

    printf("Enter a number: ");

    scanf("%d", &n);

    for (i = 1; i <= n; i++) {

        for (j = 1; j <= i; j++) {

            printf("%d ", count);

            count++;

        }

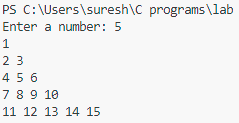
        printf("\n");

    }

    return 0;

}

1. **OUTPUT**

****

**21. Write a program in C to make such a pattern like a pyramid with numbers increased by 1.**

**1**

**2 3**

**4 5 6**

**7 8 9 10**

1. **ALGORITHM**

**STEP 1:** Start

**STEP 2:** Declare n, i=1, j=1, k=1, count = 1

**STEP 3:** Read n

**STEP 4:** If i <= n go to step 5 else go to step 8

**STEP 5:** If j <= n-i

print a white space

j++

**STEP 6:** If k <= i

print count

count++

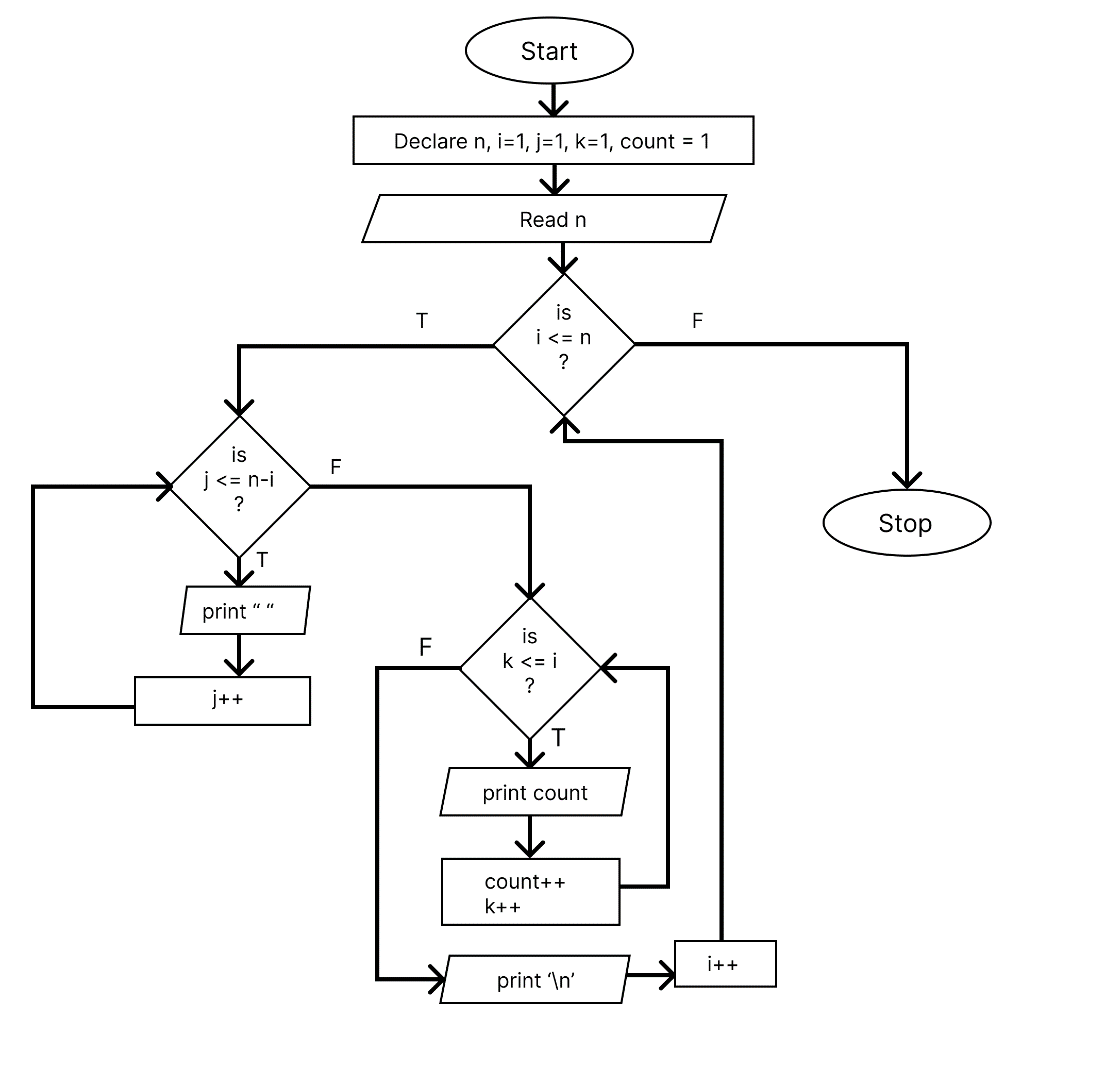
k++

**STEP 7:** Print ‘\n’

**STEP 8:** i++ and go to step 4

**STEP 9:** Stop

1. **FLOWCHART**

****

1. **PROGRAM**

#include<stdio.h>

int main() {

    int n, i, j, k, count = 1;

    printf("Enter a number: ");

    scanf("%d", &n);

    for (i = 1; i <= n; i++) {

        for (j = 1; j <= n-i; j++) {

            printf(" ");

        }

        for (k = 1; k <= i; k++) {

            printf("%d ", count);

            count++;

        }

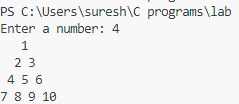
        printf("\n");

    }

    return 0;

}

1. **OUTPUT**

****

**22. Write a program in C to make such a pattern like a pyramid with an asterisk.**

**\***

**\* \***

**\* \* \***

**\* \* \* \***

**a) ALGORITHM**

**STEP 1:** Start

**STEP 2:** Declare n, i=1, j=1, k=1

**STEP 3:** Read n

**STEP 4:** If i <= n go to step 5 else go to step 8

**STEP 5:** If j <= n-i

print a white space

j++

**STEP 6:** If k <= i

print “\*”

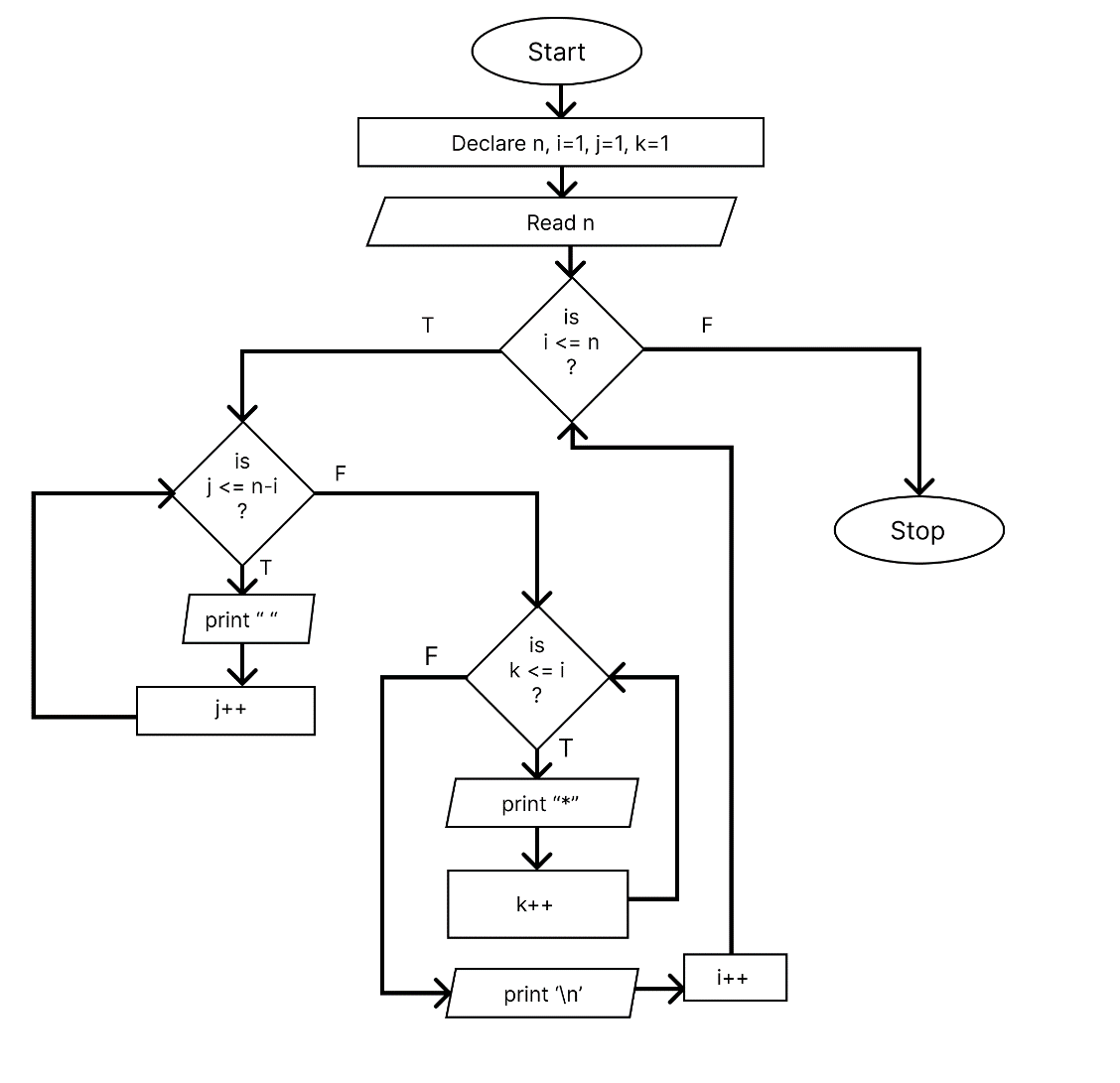
k++

**STEP 7:** Print ‘\n’

**STEP 8:** i++ and go to step 4

**STEP 9:** Stop

**b) FLOWCHART**

****

1. **PROGRAM**

#include <stdio.h>

int main() {

    int n, i, j, k;

    printf("Enter a number: ");

    scanf("%d", &n);

    for (i = 1; i <= n; i++) {

        for (j = 1; j <= n - i; j++) {

            printf(" ");

        }

        for (k = 1; k <= i; k++) {

            printf("\* ");

        }

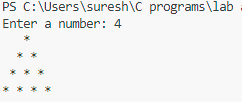
        printf("\n");

    }

    return 0;

}

**d) OUTPUT**

****