1) using if,else

```
#include<stdio.h>

int main()

{
        int marks;

        // printing statement

        printf(" enter marks under(0-100): " );

        scanf("%d",&marks);

        if(marks>90)

        {

          printf("pass");

        }

        else{

          printf("fail");

          }

        return 0;

}
```

out put:

```
enter marks under(0-100): 67

fail
```

2) if , else if statement

```
#include <stdio.h>

  int main() {

    int marks;

    printf("Enter the student's marks (0-100): ");

    // Read the marks from the user

    scanf("%d", &marks)

    if (marks >= 90 && marks <= 100) {

        printf("Grade: A\n");

    } else if (marks >= 80 && marks < 90) {

        printf("Grade: B\n");

    } else if (marks >= 70 && marks < 80) {

        printf("Grade: C\n");

    } else if (marks >= 60 && marks < 70) {

        printf("Grade: D\n");

    } else if (marks >= 0 && marks < 60) {

        printf("Grade: F (Fail)\n");

    } else {

        printf("Invalid marks entered. Please enter marks between 0 and 100.\n");

    }

    return 0;
}
```

output :

```
Enter the student's marks (0-100): 68
```

Grade: D

3) **using array**

```
#include <stdio.h>

int main() {

   int numbers[5] = {10, 20, 30, 40, 50};

   printf("Elements of the array are:\n");

   for (int i = 0; i < 5; i++) {

      printf("Element at index %d: %d\n", i, numbers[i]);
   }

   return 0;

 }
```

output:

```
Elements of the array are:

Element at index 0: 10

Element at index 1: 20

Element at index 2: 30

Element at index 3: 40

Element at index 4: 50
```

4) linear search algorithm

```
#include <stdio.h>
int main() {
    int array[100];
    int n, search_element, i;
    int found_at_index = -1;
    printf("Enter the number of elements in the array: ");
    scanf("%d", &n);
    printf("Enter %d integer(s):\n", n);
    for (i = 0; i < n; i++) {
        printf("Element %d: ", i + 1);
        scanf("%d", &array[i]);
    }
    printf("Enter the element to search: ");
    scanf("%d", &search_element);

    for (i = 0; i < n; i++) {
        if (array[i] == search_element) {
            found_at_index = i;
            break;
        }
    }

    if (found_at_index != -1) {
        printf("Element %d found at position %d (1-based indexing).\n", search_element, found_at_index + 1);
    } else {
        printf("Element %d not found in the array.\n", search_element);
    }
    return 0;

}
```

output:

```
Enter the number of elements in the array: 10

Enter 10 integer(s):

Element 1: 1

Element 2: 34

Element 3: 2

Element 4: 3

Element 5: 4

Element 6: 5

Element 7: 6

Element 8: 7
Element 9: 8

Element 10: 9

Enter the element to search: 2

Element 2 found at position 3 (1-based indexing).
```

5) string length

```
#include <stdio.h>

#include <string.h>

int main() {

    char myString[50];

    printf("Enter a string: ");

    scanf("%s", myString);

    printf("You entered: %s\n", myString);

    printf("Length of the string: %lu\n", strlen(myString));


    return 0;

}
```

output:

```
Enter a string: latha

You entered: latha

Length of the string: 5
```

6) Binary Search Algorithm

```
#include <stdio.h>
// Function to perform binary search
int binarySearch(int arr[], int size, int target) {
    int low = 0;
    int high = size - 1;
    while (low <= high) {
        int mid = low + (high - low) / 2; // Calculate middle index
        // Check if target is present at mid
        if (arr[mid] == target) {
            return mid; // Element found, return its index
        }
        // If target is greater, ignore left half
        else if (arr[mid] < target) {
            low = mid + 1;
        }
        // If target is smaller, ignore right half
        else {
            high = mid - 1;
        }
    }
    return -1; // Element not found
}
int main() {
    int arr[] = {10, 20, 30, 40, 50, 60, 70, 80, 90, 100}; // Sorted array
    int size = sizeof(arr) / sizeof(arr[0]); // Calculate array size
    int target = 60; // Element to search for
    int result = binarySearch(arr, size, target); // Call binary search function
    if (result != -1) {
        printf("Element %d found at index %d.\n", target, result);
    } else {
        printf("Element %d not found in the array.\n", target);
    }
    // Example for an element not found
    target = 55;
    result = binarySearch(arr, size, target);
    if (result != -1) {
        printf("Element %d found at index %d.\n", target, result);
    } else {
        printf("Element %d not found in the array.\n", target);
    }

return 0;
}
```

output:

```
Element 60 found at index 5.

Element 55 not found in the array.
```

7) pattern

```
#include <stdio.h>

int main() {

    int rows, i, j;

    printf("Enter the number of rows: ");

    scanf("%d", &rows);

    // Outer loop for rows

    for (i = 1; i <= rows; i++) {

        // Inner loop for columns (printing stars)

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

            printf("* ");
        }
        printf("\n"); // Move to the next line after each row
    }
    return 0;
}
```

output:

```
Enter the number of rows: 6

*
```

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

8) Reversed Number

```
#include <stdio.h>
int main() {
    int number, reversedNumber = 0, remainder;
    // Prompt the user to enter a number
    printf("Enter an integer: ");
    // Read the integer from the user
    scanf("%d", &number);
    // Loop until the number becomes 0
    while (number != 0) {
        // Get the last digit of the number
        remainder = number % 10;
        // Build the reversed number
        reversedNumber = reversedNumber * 10 + remainder;
        // Remove the last digit from the original number
        number /= 10;
    }
    // Print the reversed number
    printf("Reversed Number = %d\n", reversedNumber);

    return 0;

}

~
```

output:

```
Enter an integer: 54678

Reversed Number = 87645
```

9) Reversed String

```
#include <stdio.h>

#include <string.h> // Required for strlen()

int main() {

   char str[100]; // Declare a character array to store the string

   char temp;     // Temporary variable for swapping characters

   int i, j;      // Loop counters

   printf("Enter a string: ");

   // Using fgets to safely read input, preventing buffer overflow

   fgets(str, sizeof(str), stdin);

   // Remove the newline character if fgets reads it

   str[strcspn(str, "\n")] = 0;

     int len = strlen(str);

   // Loop to reverse the string

     // Loop continues until 'i' crosses 'j' (middle of the string)

   for (i = 0, j = len - 1; i < j; i++, j--) {

       temp = str[i];      // Store the character at the current 'i' position

       str[i] = str[j];    // Replace the character at 'i' with the character at 'j'
```

```
        str[j] = temp;      // Replace the character at 'j' with the stored character (original 'i')
    }
    printf("Reversed string: %s\n", str);
    return 0;
}
```

output:

```
Enter a string: latha rathod!
Reversed string: !dohtar ahtal
```

10) Fibonacci Series

```
#include <stdio.h>
int main() {
  int n, t1 = 0, t2 = 1, nextTerm;
  printf("Enter the number of terms: ");
  scanf("%d", &n);
  printf("Fibonacci Series: ");
  for (int i = 1; i <= n; ++i) {
    printf("%d ", t1);
    nextTerm = t1 + t2;
    t1 = t2;
    t2 = nextTerm;
  }
  printf("\n");
  return 0;
}
```

output:

```
Enter the number of terms: 7

Fibonacci Series: 0 1 1 2 3 5 8
```

11) Using Pointer

```
#include <stdio.h>
int main() {
   // 1. Declare a regular integer variable
   int number = 10;

   // 2. Declare a pointer variable that can point to an integer
   // The asterisk (*) indicates it's a pointer
   int *ptr;

   // 3. Assign the address of 'number' to 'ptr'
   // The ampersand (&) is the "address-of" operator
   ptr = &number;
```

```
    // 4. Print the value of 'number' directly
    printf("Value of number: %d\n", number);


    // 5. Print the address of 'number'
    // Use %p for printing addresses
    printf("Address of number: %p\n", &number);


    // 6. Print the value stored in 'ptr' (which is the address of 'number')
    printf("Value of ptr (address it holds): %p\n", ptr);


    // 7. Print the value 'ptr' points to (dereferencing)
    // The asterisk (*) when used with a pointer variable dereferences it,
    // meaning it gives you the value at the address the pointer holds.
    printf("Value that ptr points to: %d\n", *ptr);


    return 0;
}
```

output:

```
Value of number: 10


Address of number: 0x7ffeefbff484  // (This address will vary)


Value of ptr (address it holds): 0x7ffeefbff484 // (This address will vary)
Value that ptr points to: 10
```

12) Bitwise Operators

```
#include <stdio.h>

int main() {

    unsigned char a = 5; // Binary: 00000101

    unsigned char b = 3; // Binary: 00000011

    int result;


    printf("a = %d (Binary: 00000101)\n", a);

    printf("b = %d (Binary: 00000011)\n\n", b);


    // Bitwise AND (&)

    // Sets a bit if both corresponding bits are 1.

    result = a & b; // 00000001

    printf("Bitwise AND (a & b): %d (Binary: 00000001)\n", result);

    // Bitwise OR (|)
    // Sets a bit if at least one corresponding bit is 1.
    result = a | b; // 00000111
    printf("Bitwise OR (a | b): %d (Binary: 00000111)\n", result);
    // Bitwise XOR (^)
    // Sets a bit if the corresponding bits are different.
    result = a ^ b; // 00000110
    printf("Bitwise XOR (a ^ b): %d (Binary: 00000110)\n", result);
    // Bitwise NOT (~)
    // Inverts all bits (0 becomes 1, 1 becomes 0).
    // Note: The output for ~a will depend on the size of int, as it's a signed operation.
    // For unsigned char, ~a flips all 8 bits.
    result = ~a; // For unsigned char a=5 (00000101), ~a would be 11111010 (250)
                 // When stored in a signed int, this is interpreted differently due to two's complement.
    printf("Bitwise NOT (~a): %d (Interpretation depends on data type size and signedness)\n",
result);

    // Left Shift (<<)
    // Shifts bits to the left, filling with 0s on the right. Multiplies by powers of 2.
    result = a << 1; // 00001010 (10)
    printf("Left Shift (a << 1): %d (Binary: 00001010)\n", result);
    // Right Shift (>>)
    // Shifts bits to the right. For unsigned types, fills with 0s on the left. Divides by powers of 2.
    result = b >> 1; // 00000001 (1)
    printf("Right Shift (b >> 1): %d (Binary: 00000001)\n", result);
    return 0;
}
```

output:

```
a = 5 (Binary: 00000101)

b = 3 (Binary: 00000011)

Bitwise AND (a & b): 1 (Binary: 00000001)

Bitwise OR (a | b): 7 (Binary: 00000111)

Bitwise XOR (a ^ b): 6 (Binary: 00000110)

Bitwise NOT (~a): -6 (Interpretation depends on data type size and signedness)

Left Shift (a << 1): 10 (Binary: 00001010)

Right Shift (b >> 1): 1 (Binary: 00000001)
```

13) Finding Largest of Three Numbers

```
#include <stdio.h>

int main() {

    double n1, n2, n3;

    printf("Enter three numbers: ");

    scanf("%lf %lf %lf", &n1, &n2, &n3);


    if (n1 >= n2 && n1 >= n3) {

        printf("%.2lf is the largest number.\n", n1);

    } else if (n2 >= n1 && n2 >= n3) {

        printf("%.2lf is the largest number.\n", n2);

    } else {

        printf("%.2lf is the largest number.\n", n3);
    }
    return 0;
 }
```

output:

```
 Enter three numbers: 10.5 25.2 8.9

 25.20 is the largest number.
```

14) Finding Factorial of a Number

```
#include <stdio.h>
 int main() {
    int n, i;
    unsigned long long factorial = 1; // Use unsigned long long for larger factorials
    printf("Enter a non-negative integer: ");
    scanf("%d", &n);

    if (n < 0) {
       printf("Factorial of a negative number doesn't exist.\n");
    } else {
       for (i = 1; i <= n; ++i) {
          factorial *= i;
```

```
        }
        printf("Factorial of %d = %llu\n", n, factorial);
    }
    return 0;
}
```

output:

```
Enter a non-negative integer: 5


Factorial of 5 = 120
```

15) Finding palindromes.

```
#include <stdio.h>
int main() {
    int n, original_n, reversed_n = 0, remainder;
    printf("Enter an integer: ");
    scanf("%d", &n);
    original_n = n; // Store the original number

    while (n != 0) {
        remainder = n % 10;
        reversed_n = reversed_n * 10 + remainder;
        n /= 10;
    }
    if (original_n == reversed_n) {
        printf("%d is a palindrome.\n", original_n);
    } else {
        printf("%d is not a palindrome.\n", original_n);
    }
    return 0;
}
```

output:

```
Enter an integer: 121
121 is a palindrome.
```

16) Sum of Digits

```
#include <stdio.h>

int main() {
    int n, sum = 0, remainder;

    printf("Enter an integer: ");
    scanf("%d", &n);

    while (n != 0) {
        remainder = n % 10;
        sum += remainder;
        n /= 10;
    }

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

    return 0;
}
```

output:

```
Enter an integer: 1234
Sum of digits = 10
```

17) Palindrome String Checker

```
#include <stdio.h>
#include <string.h>
int main() {
    char str[100];
    int i, length;
    int isPalindrome = 1;
    printf("Enter a string: ");
    scanf("%s", str);
    length = strlen(str);
    for (i = 0; i < length / 2; i++) {
        if (str[i] != str[length - 1 - i]) {

            isPalindrome = 0; // Not a palindrome

            break;

        }

    }

    if (isPalindrome == 1) {

        printf("%s is a palindrome.\n", str);

    } else {

        printf("%s is not a palindrome.\n", str);

    }

    return 0;

}
```

output:

```
Enter a string: madam

madam is a palindrome.
```

18) Finding Armstrong numbers

```
#include <stdio.h>


#include <math.h>


int main() {


    int start, end, num, temp, remainder, n = 0;

    double result = 0.0;
```

```
printf("Enter start and end values: ");
scanf("%d %d", &start, &end);
printf("Armstrong numbers between %d and %d are: ", start, end);
for (num = start; num <= end; num++) {
    temp = num;
    n = 0;
    result = 0.0;
    // Count digits

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

    temp = num;
    // Calculate Armstrong sum
    while (temp != 0) {
        remainder = temp % 10;
        result += pow(remainder, n);
        temp /= 10;
    }
    if ((int)result == num) {

        printf("%d ", num);
    }
    }

    return 0;

}
```

output:

```
Enter start and end values: 100 500

Armstrong numbers between 100 and 500 are: 153 370 371 407
```

19) Tower of Hanoi(Recursive)

```
#include <stdio.h>

void towerOfHanoi(int n, char from, char to, char aux) {

    if (n == 1) {

        printf("Move disk 1 from %c to %c\n", from, to);

        return;

    }

    towerOfHanoi(n - 1, from, aux, to);

    printf("Move disk %d from %c to %c\n", n, from, to);

    towerOfHanoi(n - 1, aux, to, from);

}
```

```
int main() {
    int n;
    printf("Enter number of disks: ");
    scanf("%d", &n);
    towerOfHanoi(n, 'A', 'C', 'B');
    return 0;
}
```

output:

```
Enter number of disks: 3
Move disk 1 from A to C
Move disk 2 from A to B
Move disk 1 from C to B
Move disk 3 from A to C
Move disk 1 from B to A
Move disk 2 from B to C
Move disk 1 from A to C
```

20) Sum of Digits Using a Function

```
#include <stdio.h>
int fibonacci(int n) {
    if (n == 0) return 0;
    if (n == 1) return 1;
    return fibonacci(n - 1) + fibonacci(n - 2);
}


int main() {
    int terms, i;
```

```c
    printf("Enter number of terms: ");
    scanf("%d", &terms);

    printf("Fibonacci Series: ");
    for (i = 0; i < terms; i++) {
        printf("%d ", fibonacci(i));
    }
    return 0;
}
```

```
Enter number of terms: 6
Fibonacci Series: 0 1 1 2 3 5
```

```c
#include <stdio.h>

// Function to calculate sum of digits
int sumOfDigits(int num) {
    int sum = 0, digit;
    while (num > 0) {
        digit = num % 10;
        sum += digit;
        num /= 10;
    }
    return sum;
}

int main() {
    int number, result;
    printf("Enter a number: ");

    scanf("%d", &number);


    result = sumOfDigits(number);


    printf("Sum of digits of %d is %d\n", number, result);

    return 0;

}
```

output:

```
Enter a number: 12345


Sum of digits of 12345 is 15
```