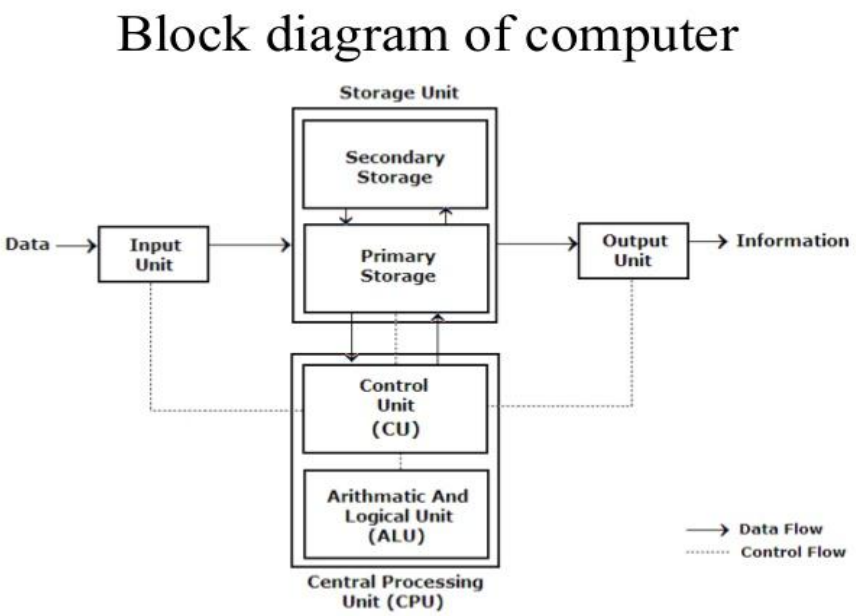
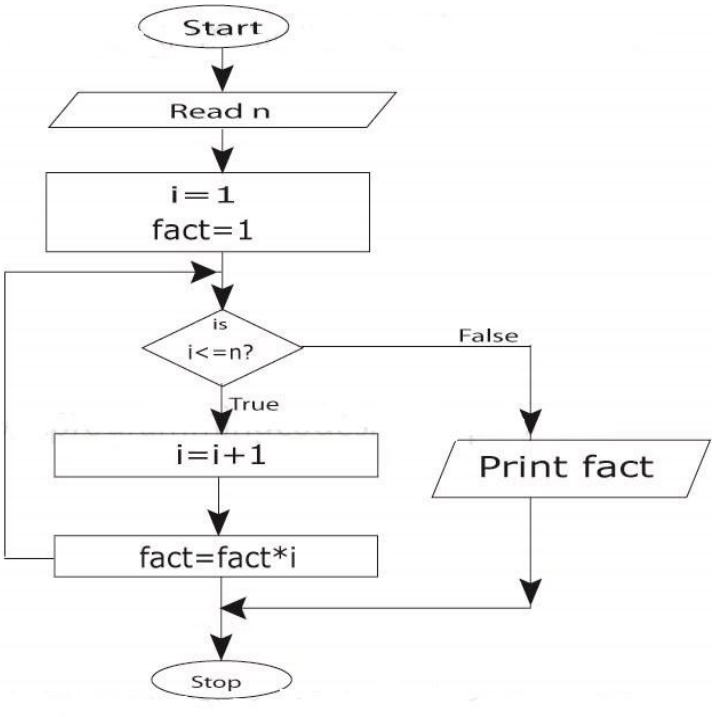
1) **Solution:-** Block diagram of digital computer-



2) **Solution:**- Flowchart to find the factorial of a number-



3) **Solution:**- Compilation error. Here, there is an error in the statement a+b=2+b; The left part of assignment operator should have only one variable.

4) **Solution:**- n%2==0? printf(“is even”) : printf(“is odd”);

5) a. **Solution:**-

Difference Between break and continue

break	continue
A break can appear in both switch and loop (for, while, do) statements.	A continue can appear only in loop (for, while, do) statements.
A break causes the switch or loop statements to terminate the moment it is executed. Loop or switch ends abruptly when break is encountered.	A continue doesn't terminate the loop, it causes the loop to go to the next iteration. All iterations of the loop are executed even if continue is encountered. The continue statement is used to skip statements in the loop that appear after the continue.
The break statement can be used in both switch and loop statements.	The continue statement can appear only in loops. You will get an error if this appears in switch statement.
When a break statement is encountered, it terminates the block and gets the control out of the switch or loop.	When a continue statement is encountered, it gets the control to the next iteration of the loop.
A break causes the innermost enclosing loop or switch to be exited immediately.	A continue inside a loop nested within a switch causes the next loop iteration.

5) b. **Solution: -** Two difference between for & do while loop in c

For Loop-

1. It executes only when the condition is TRUE, otherwise it does not executes.
2. for-loops are counter-controlled**,** meaning that they are normally used whenever the number of iterations is known in advance.

Do-While loop-

1. Do-While loop, execute the statements in the loop first before checks for the condition. At least one iteration takes places, even if the condition is false.
2. This guarantees that the loop will be performed at least once, which is useful for checking user input among other things.

Example- Syntax:

```
do {
    body;
} while( condition );
```

6) a). **Solution:-** 1 1 2 3 5 8 13

```
fib1=0; fib2=1;
    for(i=1;i<=7;i++)
    {
        fib=fib1+fib2;
            fib2=fib1;
            fib1=fib;
          printf("%d",fib);
    }
```

b) **Solution:-** 8 27 64 125

```
  for(i=2;i<=5,i++)
    {
            j=pow(i,3);
            printf("%d",j);
    }
```

7) **Solution:-**

```
#include<stdio.h>
 main() {
 int i, j, mat[10][10], row, col;
 int sum = 0;
printf("\nEnter the number of Rows : ");
 scanf("%d", &row);
 printf("\nEnter the number of Columns : ");
 scanf("%d", &col);
//Accept the Elements in m x n Matrix
 for (i = 0; i < row; i++) {
for (j = 0; j < col; j++) {
printf("\nEnter the Element a[%d][%d] : ", i, j);
scanf("%d", &mat[i][j]);
```

```
 }
}
//Addition of all Diagonal Elements
for (i = 0; i < row; i++) {
for (j = 0; j < col; j++) {
 if (i == j)
sum = sum + mat[i][j];
}
}
//Print out the Result
printf("\nSum of Diagonal Elements in Matrix : %d", sum);
}
```

8) a. **Solution:**-

Declaring Arrays

To declare an array in C, a programmer specifies the type of the elements and the number of elements required by an array as follows − type arrayName [arraySize];

double balance[10];

Here balance is a variable array which is sufficient to hold up to 10 double numbers.

Initializing Arrays

You can initialize an array in C either one by one or using a single statement as follows −

double balance[5] = {1000.0, 2.0, 3.4, 7.0, 50.0};

8) b **Solution:**-

The following declaration and initialization create a string consisting of the word "Hello". To hold the null character at the end of the array, the size of the character array containing the string is one more than the number of characters in the word "Hello."

char greeting[6] = {'H', 'e', 'l', 'l', 'o', '\0'};

If you follow the rule of array initialization then you can write the above statement as follows −

char greeting[] = "Hello";

Q.8.c **Solution:**-

```
#include<stdio.h>
int main()
{
        char name[50],number_of_vowels=0;
        printf(“Enter a name:”);
        scanf(“%s”, name);
        for(i=0l; name[i]!=’\0’;i++)
        {
                If(name[i]==’a’ || name[i]==’A’ || name[i]==’e’ || name[i]==’E’ || name[i]==’i’ ||
name[i]==’I’ || name[i]==’o’ || name[i]==’O’ || name[i]==’u’ || name[i]==’U’ )
                        number_of_vowels++;
        }
        printf(“Number of vowels in the string is :%d”, number_of_vowels);
        return 0;
}
```