Practical 1

Aim: Practice using scratch programming/snap programming.

Code:

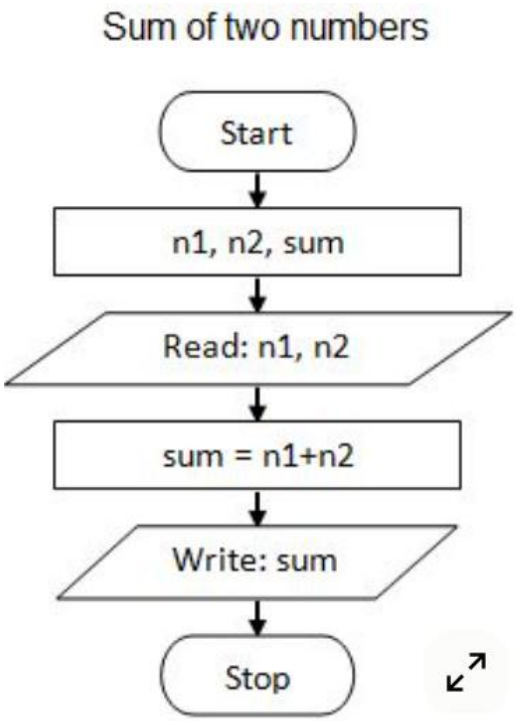
1. **Open Scratch**
   - Go to scratch.mit.edu.
   - Click **Create** (top menu).
     You'll see the stage (right side), sprite (cat), and blocks (left side).
2. **Understand the Interface**
   - **Stage (top right):** Where your program runs.
   - **Sprites (bottom right):** Characters/objects in your project.
   - **Blocks (left):** Drag-and-drop code blocks.
     - Motion (blue) → Move, turn
     - Looks (purple) → Say, costumes
     - Sound (pink) → Play sounds
     - Events (yellow) → Start with clicks/keys
     - Control (orange) → Loops, if statements

3. **First Program (Hello World)**
   a) From **Events**, drag when green flag clicked.
   b) From **Looks**, drag say "Hello!" for 2 seconds under it.
   c) Click the green flag ▶️ — the cat says "Hello!"
4. **Add Movement**
   a) Drag when green flag clicked.
   b) Add move 10 steps.
   c) Add a **Control → forever** loop around it.
   d) Add if on edge, bounce.
   e) Use Keyboard Input
   1. Drag when [right arrow] key pressed.
   2. Add change x by 10.
   3. Duplicate, but change:
      - left arrow → change x by -10
      - up arrow → change y by 10
      - down arrow → change y by -10

Practical 2

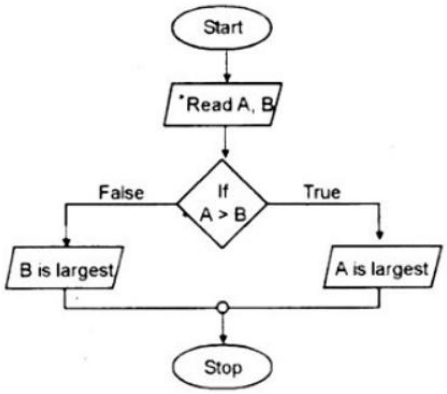
Aim: Design and develop various problem statement using flowchart and Algorithm.

1. **Sum of 2 given numbers**



1. **Start**
2. **Input A, B**
3. **Compute SUM = A + B**
4. **Print SUM**
5. **Stop**

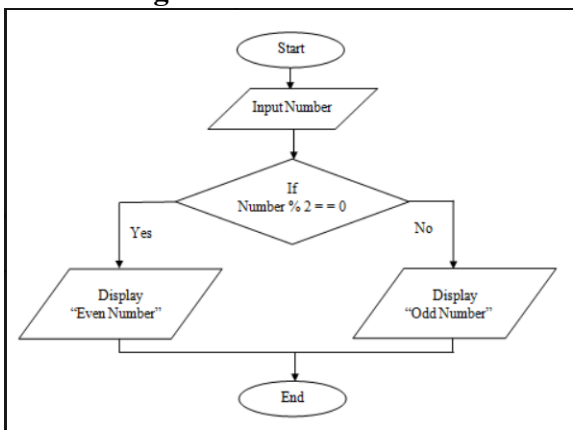
2. **Find max of 2 numbers**



Algorithm:

1. Start
2. Input A, B
3. If A > B
   - Print “A is greater”
     Else
   - Print “B is greater”
4. Stop

3. Find if given number is odd or even



Algorithm

1. Start
2. Input N
3. Compute R = N % 2
4. If R == 0 → Print “Even”
   Else → Print “Odd”
5. Stop

Practical 3

Aim: Design and test C programs using constants, variables, data types and different operators.

Code .

```
#include <stdio.h>
int main() {
    // Constants
    const float PI = 3.14159;

    // Variables
    int a = 10, b = 3;
    float result;

    // Arithmetic operators
    result = a + b;
    printf("Addition: %d + %d = %.2f\n", a, b, result);

    result = a - b;
    printf("Subtraction: %d - %d = %.2f\n", a, b, result);

    result = a * b;
    printf("Multiplication: %d * %d = %.2f\n", a, b, result);

    result = (float)a / b;
    printf("Division: %d / %d = %.2f\n", a, b, result);

    result = a % b;
    printf("Modulus: %d %% %d = %.2f\n", a, b, result);

    // Relational operators
    printf("a > b: %d\n", a > b);
    printf("a < b: %d\n", a < b);
    printf("a == b: %d\n", a == b);
    printf("a != b: %d\n", a != b);

    // Logical operators
    printf("(a > 5 && b < 5): %d\n", (a > 5 && b < 5));
    printf("(a < 5 || b < 5): %d\n", (a < 5 || b < 5));
    printf("!(a == b): %d\n", !(a == b));

    // Using constant
    printf("Value of PI constant: %.5f\n", PI);
    return 0;
}
```

Output:
Addition: 10 + 3 = 13.00
Subtraction: 10 - 3 = 7.00
Multiplication: 10 * 3 = 30.00
Division: 10 / 3 = 3.33
Modulus: 10 % 3 = 1.00
a > b: 1
a < b: 0
a == b: 0
a != b: 1
(a > 5 && b < 5): 1
(a < 5 || b < 5): 1
!(a == b): 1
Value of PI constant: 3.14159

Practical 4

Aim: Design and test C programs to show formatted and unformatted input and output.

Code:

```
#include <stdio.h>
int main() {
   int age;
   char name[50];
   float salary;
   // Formatted Input
   printf("Enter your name: ");
   scanf("%s", name);  // formatted input for string
   printf("Enter your age: ");
   scanf("%d", &age);  // formatted input for integer
   printf("Enter your salary: ");
   scanf("%f", &salary);  // formatted input for float
   // Formatted Output
   printf("\n--- Formatted Output ---\n");
   printf("Name: %s\n", name);
   printf("Age: %d\n", age);
   printf("Salary: %.2f\n", salary);
   printf("Age: ");
   putchar(age + '0');  // convert single digit integer to character (works if age < 10)
   printf("\n");
   printf("Salary: ");
   printf("%f\n", salary);  // unformatted floating point using default format
   return 0;
}
```

Output:

```
Enter your name: John
Enter your age: 25
Enter your salary: 5000.50
--- Formatted Output ---
Name: John
Age: 25
Salary: 5000.50
--- Unformatted Output ---
Name:
John
Age: 2
Salary: 5000.500000
```

Practical 5

Aim: Design and test at least one C programs using below given decision making statements: (1) Simple if (2) if…else (3) Nested if (4) if…else ladder (5) switch (6) goto

Code

```
#include <stdio.h>

int main() {
   int num, choice;

   // 1. Simple if
   printf("Enter a number for simple if check (>0): ");
   scanf("%d", &num);
   if (num > 0) {
      printf("Number is positive.\n");
   }

   // 2. if…else
   printf("Enter a number for if-else check: ");
   scanf("%d", &num);
   if (num % 2 == 0) {
      printf("Number is even.\n");
   } else {
      printf("Number is odd.\n");
   }

   // 3. Nested if
   printf("Enter a number for nested if check: ");
   scanf("%d", &num);
   if (num != 0) {
      if (num > 0) {
         printf("Number is positive and non-zero.\n");
      } else {
         printf("Number is negative.\n");
      }
   } else {
      printf("Number is zero.\n");
   }

   // 4. if…else ladder
   printf("Enter a number (1-3) for if-else ladder: ");
   scanf("%d", &num);
   if (num == 1) {
      printf("You entered One.\n");
   } else if (num == 2) {
      printf("You entered Two.\n");
   } else if (num == 3) {
      printf("You entered Three.\n");
```

```
    } else {
        printf("Number is out of range.\n");
    }

    // 5. switch
    printf("Enter a number (1-3) for switch case: ");
    scanf("%d", &num);
    switch (num) {
        case 1:
            printf("Switch: One\n");
            break;
        case 2:
            printf("Switch: Two\n");
            break;
        case 3:
            printf("Switch: Three\n");
            break;
        default:
            printf("Switch: Out of range\n");
    }
    // 6. goto
    printf("Enter a number (1-3) for goto demonstration: ");
    scanf("%d", &num);
    if (num < 1 || num > 3) {
        goto error;
    }
    printf("You entered %d correctly.\n", num);
    return 0;
error:
    printf("Error: Invalid input, out of range.\n");
    return 1;
}
```

Output:

```
Enter a number for simple if check (>0): 5
Number is positive.
Enter a number for if-else check: 4
Number is even.
Enter a number for nested if check: -2
Number is negative.
Enter a number (1-3) for if-else ladder: 2
You entered Two.
Enter a number (1-3) for switch case: 3
Switch: Three
Enter a number (1-3) for goto demonstration: 5
Error: Invalid input, out of range.
```

Practical 6

Aim: Design and test C programs using the for, while and do. While loop.

Code

```
#include <stdio.h>
int main() {
   int i, n;
   // 1. For loop
   printf("For Loop: Enter a number n to print numbers 1 to n: ");
   scanf("%d", &n);
   printf("Numbers using for loop: ");
   for (i = 1; i <= n; i++) {
      printf("%d ", i);
   }
   printf("\n");

   // 2. While loop
   printf("While Loop: Enter a number n to print numbers 1 to n: ");
   scanf("%d", &n);
   i = 1;
   printf("Numbers using while loop: ");
   while (i <= n) {
      printf("%d ", i);
      i++;
   }
   printf("\n");

   // 3. Do-While loop
   printf("Do-While Loop: Enter a number n to print numbers 1 to n: ");
   scanf("%d", &n);
   i = 1;
   printf("Numbers using do-while loop: ");
   do {
      printf("%d ", i);
      i++;
   } while (i <= n);
   printf("\n");

   return 0;
}
```

Output:

```
For Loop: Enter a number n to print numbers 1 to n: 5
Numbers using for loop: 1 2 3 4 5
While Loop: Enter a number n to print numbers 1 to n: 3
Numbers using while loop: 1 2 3
Do-While Loop: Enter a number n to print numbers 1 to n: 4
Numbers using do-while loop: 1 2 3 4
```

Practical 7

Aim:Design and test a C program using break and continue statements.

Code

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

   printf("Enter a number n: ");
   scanf("%d", &n);

   printf("Demonstrating break:\n");
   for (i = 1; i <= n; i++) {
      if (i == 5) {
         printf("Break encountered at i = %d\n", i);
         break;  // exit loop when i is 5
      }
      printf("%d ", i);
   }
   printf("\n");

   printf("Demonstrating continue:\n");
   for (i = 1; i <= n; i++) {
      if (i % 2 == 0) {
         continue;  // skip even numbers
      }
      printf("%d ", i);
   }
   printf("\n");

   return 0;
}
```

Output:

```
Enter a number n: 7
Demonstrating break:
1 2 3 4 Break encountered at i = 5
Demonstrating continue:
1 3 5 7
```

Practical 8

Aim:Design and test C programs using one dimensional array and two dimensional arrays.

Code:

```
#include <stdio.h>
int main() {
   int i, j, n, rows, cols;

   // One-dimensional array
   printf("Enter the size of 1D array: ");
   scanf("%d", &n);
   int arr1D[n];

   printf("Enter %d elements for 1D array:\n", n);
   for (i = 0; i < n; i++) {
      scanf("%d", &arr1D[i]);
   }

   printf("1D Array elements are: ");
   for (i = 0; i < n; i++) {
      printf("%d ", arr1D[i]);
   }
   printf("\n");

   // Two-dimensional array
   printf("Enter number of rows and columns for 2D array: ");
   scanf("%d %d", &rows, &cols);
   int arr2D[rows][cols];

   printf("Enter elements for 2D array:\n");
   for (i = 0; i < rows; i++) {
      for (j = 0; j < cols; j++) {
         scanf("%d", &arr2D[i][j]);
      }
   }

   printf("2D Array elements are:\n");
   for (i = 0; i < rows; i++) {
      for (j = 0; j < cols; j++) {
         printf("%d ", arr2D[i][j]);
      }
      printf("\n");
   }

   return 0;
}
```

Output:

```
Enter the size of 1D array: 3
```

Enter 3 elements for 1D array:
5
10
15
1D Array elements are: 5 10 15
Enter number of rows and columns for 2D array: 2 3
Enter elements for 2D array:
1 2 3
4 5 6
2D Array elements are:
1 2 3
4 5 6

Practical 9

Aim: Design and test C programs using pointers.

Code

```
#include <stdio.h>
int main() {
   int a = 10;
   int b = 20;
   int *ptrA, *ptrB, sum;

   // Pointer initialization
   ptrA = &a;
   ptrB = &b;

   printf("Address of a: %p, Value: %d\n", ptrA, *ptrA);
   printf("Address of b: %p, Value: %d\n", ptrB, *ptrB);

   // Using pointers to modify values
   *ptrA = *ptrA + 5;
   *ptrB = *ptrB + 10;

   printf("Modified values using pointers: a = %d, b = %d\n", a, b);

   // Sum using pointers
   sum = *ptrA + *ptrB;
   printf("Sum of a and b using pointers: %d\n", sum);

   // Pointer to pointer
   int **ptrPtr = &ptrA;
   printf("Pointer to pointer points to value: %d\n", **ptrPtr);

   return 0;
}
```

Output:

```
Address of a: 0x7ffee3f6a99c, Value: 10
Address of b: 0x7ffee3f6a998, Value: 20
Modified values using pointers: a = 15, b = 30
Sum of a and b using pointers: 45
Pointer to pointer points to value: 15
```

Practical 10

Aim: Design and C programs using user defined Functions

Code

```
#include <stdio.h>

// Function prototypes
int add(int x, int y);
int multiply(int x, int y);
void greet(char name[]);
int main() {
   int a, b;
   char name[50];

   // User input
   printf("Enter your name: ");
   scanf("%s", name);
   greet(name);

   printf("Enter two numbers: ");
   scanf("%d %d", &a, &b);
   // Using functions
   printf("Sum: %d + %d = %d\n", a, b, add(a, b));
   printf("Product: %d * %d = %d\n", a, b, multiply(a, b));
   return 0;
}
// Function to add two numbers
int add(int x, int y) {
   return x + y;
}
// Function to multiply two numbers
int multiply(int x, int y) {
   return x * y;
}
// Function to greet user
void greet(char name[]) {
   printf("Hello, %s!\n", name);
}
```

Output:

```
Enter your name: Alice
Hello, Alice!
Enter two numbers: 5 7
Sum: 5 + 7 = 12
Product: 5 * 7 = 35
```

Practical 11

Aim: Design and test recursion function.

Code

```
#include <stdio.h>

// Recursive function to calculate factorial
int factorial(int n) {
    if (n == 0 || n == 1)
        return 1;
    else
        return n * factorial(n - 1);
}

int main() {
    int n;
    printf("Enter a number to calculate factorial: ");
    scanf("%d", &n);

    printf("Factorial of %d is %d\n", n, factorial(n));

    return 0;
}
```

Output:

```
Enter a number to calculate factorial: 5
Factorial of 5 is 120
```

Practical 12

Aim: Design and test a C program to test various inbuilt string functions.

Code

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

int main() {
   char str1[50], str2[50], str3[50];

   // Input strings
   printf("Enter first string: ");
   scanf("%s", str1);
   printf("Enter second string: ");
   scanf("%s", str2);

   // strlen: string length
   printf("Length of '%s' is %lu\n", str1, strlen(str1));

   // strcpy: copy string
   strcpy(str3, str1);
   printf("Copied string: %s\n", str3);

   // strcat: concatenate strings
   strcat(str1, str2);
   printf("Concatenated string: %s\n", str1);

   return 0;
}
```

Output:

```
Enter first string: Hello
Enter second string: World
Length of 'Hello' is 5
Copied string: Hello
Concatenated string: HelloWorld
```

Practical 13

Aim:Design and test C programs using file operations

Code

```
#include <stdio.h>

int main() {
    FILE *fp;
    char filename[50], ch;

    // Input filename
    printf("Enter filename to create and write: ");
    scanf("%s", filename);

    // Open file for writing
    fp = fopen(filename, "w");
    if (fp == NULL) {
        printf("Error opening file!\n");
        return 1;
    }

    // Write to file
    fprintf(fp, "This is a sample text written to the file.\n");
    fclose(fp);
    printf("Data written to file successfully.\n");

    // Open file for reading
    fp = fopen(filename, "r");
    if (fp == NULL) {
        printf("Error opening file!\n");
        return 1;
    }
    printf("Contents of the file:\n");
    while ((ch = fgetc(fp)) != EOF) {
        putchar(ch);
    }
    fclose(fp);

    return 0;
}
```

Output

```
Enter filename to create and write: sample.txt
Data written to file successfully.
Contents of the file:
This is a sample text written to the file.
```

Practical 14

Aim: Design and test C programs using Command line arguments

Code

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

int main(int argc, char *argv[]) {
   if (argc != 3) {
      printf("Usage: %s <num1> <num2>\n", argv[0]);
      return 1;
   }

   int num1 = atoi(argv[1]);  // convert string to integer
   int num2 = atoi(argv[2]);

   printf("Number 1 = %d\n", num1);
   printf("Number 2 = %d\n", num2);
   printf("Sum = %d\n", num1 + num2);
   printf("Product = %d\n", num1 * num2);

   return 0;
}
```

Output

```
$ gcc program.c -o program
$ ./program 5 7
Number 1 = 5
Number 2 = 7
Sum = 12
Product = 35
```