ASSIGNMENT-2

Q1.. 1. While the goto statement can be used to control program flow, it's generally recommended to use selection structures and repetition control structures. What are the advantages of these of over simple goto statement.

Ans: The use of selection structures (like `if`, `switch`) and repetition control structures (such as `for`, `while`, `do-while`) is preferred over the `goto` statement for several reasons, which include:
Readability: Clear and logical flow, easier to understand.

Maintainability: Structured programming reduces complexity.

Avoids Spaghetti Code: Prevents tangled and unpredictable jumps.
Error Reduction: Reduces bugs caused by arbitrary jumps.
Debugging: Easier to debug due to structured flow.

Q2 . State the purpose of the break statement within a switch case. Can a switch case work without it ? Explain your choice with an example.

Ans: The `break` statement prevents fall-through by terminating a case after execution. Without it, control continues to subsequent cases until a `break` or the end of the block.

Yes, a `switch` case can technically work without `break`. In such cases, execution will "fall through" to the next case(s) until a `break` or the end of the `switch` block is encountered. Whether or not to use `break` depends on the intended logic.

Q3. Describe the functions of break and continue statements within loops. Illustrate their applications in various programming contexts.

Ans: The break statement is used to terminate the execution of a loop immediately. Once break is encountered, control exits the loop and continues with the next statement after the loop. The continue statement skips the remaining statements in the current iteration of the loop and moves to the next iteration.

Q4. Under what circumstances should you employ if-else statements, switch statements, or loops? What criteria should be evaluated when selecting the appropriate control flow structure?

Ans:

Structure	When to Use	Advantages	Disadvantages
`if-else`	For complex conditions or ranges.	Flexible, handles complex logic.	Can become messy with many conditions.
`switch`	For discrete values of a variable.	Cleaner for multiple cases, better readability.	Limited to discrete, simple comparisons.
Loops	For repetitive tasks or iterating through data.	Automates repetition, saves code.	Risk of infinite loops if not handled correctly.

Q5. Find the Output:

```c
#include<stdio.h>
int main(){
    float x = 25.0, y = 10.0;
    if(y != (x - 10.0)) // Condition: y (10.0) != x - 10.0 (15.0), evaluates to true
        x = x - 10.0;   // Executes: x = 25.0 - 10.0 = 15.0
    else
        x = x / 2.0;
    return 0;}
```

```c
#include<stdio.h>
int main(){
    float x = 25.0, y = 10.0;
    if(y < 15.0)          // Condition: y (10.0) < 15.0, evaluates to true
        if(y >= 0.0)      // Nested condition: y (10.0) >= 0.0, evaluates to true
            x = 5 * y;    // Executes: x = 5 * 10.0 = 50.0
        else
            x = 2 * y;    // Skipped as y >= 0.0 is true
    else
        x = 3 * y;        // Skipped as y < 15.0 is true
    return 0;
}
```

```c
#include<stdio.h>
int main(){
    int i = 0;
    while (i <= 5) {           // Loop runs while i <= 5
        printf("%3d %3d\n", i, 10 - i);  // Prints i and (10 - i)
        i = i + 1;             // Increment i by 1
    }
    return 0;
}

/* Output:
  0  10
  1   9
  2   8
  3   7
  4   6
  5   5
*/
```

```c
#include<stdio.h>
int main(){
    int i = 1;                 // Initializing i to 1
    while (1) {                // Infinite loop
        printf("%d ", i++);    // Prints i and increments it
        if (i > 10)            // Breaks when i > 10
            break;
    }
    return 0;
}

/* Output:
1 2 3 4 5 6 7 8 9 10
*/
```

```c
#include<stdio.h>
int main() {
    int i, j, n = 5;
    for (i = 1, j = 1; j <= n; i += 2, j++) {  // i increments by 2, j increments by 1
        printf("%d%d\n", i, j);  // Prints i and j
    }
    return 0;
}

/* Output:
11
32
53
74
95
*/
```

```c
#include<stdio.h>
int main() {
    int count = 11;
    while (--count + 1);  // Decrements count until count becomes 0
    printf("count down is %d \n", count);
    return 0;
}

/* Output:
count down is 0
*/
```

```c
#include<stdio.h>
int main() {
    int m, n;
    for (m = 9; m > 0; --m)          // Outer loop runs 9 times
        for (n = 6; n > 1; --n)      // Inner loop runs 5 times
            printf("#########\n");   // Prints 45 times (9 * 5)
    return 0;
}

/* Output:
#########
#########
#########
#########
#########
... (repeated 45 times)
*/
```

```c
#include<stdio.h>
int main(){
    int i = 2;
    switch(i) {
        default: printf("Hello ");  // Default is executed first
        case 1: printf("Hello ");  // No break, so fall-through occurs
        case 2:
        case 3: printf("Hello ");  // Executes case 2 and 3
    }
    return 0;
}

/* Output:
Hello Hello Hello
*/
```

```c
#include <stdio.h>
int main() {
    int i = 0;
    while(i++) {                // First, i is incremented after checking the condition
        printf("%d ", i);         // Prints the value of i
        if (i > 2)                // If i is greater than 2, breaks the loop
            break;
    }
    return (0);
}
```

```c
#include<stdio.h>
int main() {
    int a = 10;
    if(a = 0) {                 // Assignment instead of comparison, a is assigned 0
        printf("%d %d", sizeof(2.3f), sizeof(2.3));  // This won't execute
    }
    return(0);
}
```

```c
/*Write a program that determines the day number (1 to 366) in a year for a date that is
provided as input data. As an example, January 1, 1994, is day 1. December 31, 1993, is
day 365. December 31, 1996, is day 366, since 1996 is a leap year. A year is a leap year
if it is divisible by four, except that any year divisible by 100 is a leap year only if
it is divisible by 400. Your program should accept the month, day, and year as integers.
Include a function leap that returns 1 if called with a leap year, 0 otherwise*/

#include <stdio.h>

int main() {
    int day, month, year;
    printf("Enter the date (MM DD YYYY): ");
    scanf("%d %d %d", &month, &day, &year);
```

```c
    int days_in_month[] = { 31, 28, 31, 30, 31, 30, 31, 31, 30, 31, 30, 31 };
    if ((year % 4 == 0 && year % 100 != 0) || (year % 400 == 0)) {
        days_in_month[1] = 29;
    }
    int day_num = 0;
    for (int i = 0; i < month - 1; i++) {
        day_num += days_in_month[i];
    }
    day_num += day;

    printf("The day number for %02d/%02d/%d is: %d\n", month, day, year, day_num);
    return 0;
}
```

```c
/* 7. Write a program to calculate the grade of a student using switch case. The program
should ask the user about the marks obtained by the student and find the grade according
to following rule if mark ≥ 95the grade ‘O’, if 81 ≤ mark ≤ 94 then grade ‘A’, if 71 ≤
mark ≤ 80 then grade ‘B’, if 61 ≤ mark ≤70thengrade ‘C’, if 51 ≤ mark ≤ 60 then grade ‘D’,
if 40 ≤ mark ≤ 50 then grade ‘E’, if mark < 40 then grade ‘F’.*/
#include <stdio.h>

int main() {
    int marks;
    char grade;
    printf("Enter the marks obtained by the student: ");
    scanf("%d", &marks);
    switch (marks / 10) {
        case 10:
        case 9: grade = 'O'; break;
        case 8: grade = 'A'; break;
        case 7: grade = 'B'; break;
        case 6: grade = 'C'; break;
        case 5: grade = 'D'; break;
        case 4: grade = 'E'; break;
        default: grade = 'F'; break;
    }
    printf("The grade of the student is: %c\n", grade);
    return 0;
}
```

The natural logarithm can be approximated by the following series

$$\frac{x-1}{x}+\frac{1}{2}\left(\frac{x-1}{x}\right)^2+\frac{1}{2}\left(\frac{x-1}{x}\right)^3+\frac{1}{2}\left(\frac{x-1}{x}\right)^4+\ldots$$

Write a program that accepts x as an input through the keyboard and calculates the sum of first nine terms of this series.

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

```c
int main() {
    double x, term, sum = 0.0;
    int i;
    printf("Enter the value of x: ");
    scanf("%lf", &x);
    for (i = 1; i <= 9; i++) {
        term = pow((x - 1) / x, i) / i;
        sum += term;
    }
    printf("The sum of the first nine terms of the series is: %.6lf\n", sum);
    return 0;
}
```

9. Design a C program to display the following pattern based on the input given by the user.

```
Enter the choice of the character : G

A B C D E F G F E D C B A
A B C D E F   F E D C B A
A B C D E       E D C B A
A B C D           D C B A
A B C               C B A
A B                   B A
A                       A
```

```c
#include <stdio.h>

int main() {
    char ch;
    printf("Enter the choice of the character: ");
    scanf(" %c", &ch);
    int rows = ch - 'A' + 1;
    for (int i = rows; i >= 1; i--) {
        for (char c = 'A'; c <= 'A' + i - 1; c++) {
            printf("%c ", c);
        }
        for (char c = 'A' + i - 2; c >= 'A'; c--) {
            printf("%c ", c);
        }
        printf("\n");
    }
    for (int i = 1; i < rows; i++) {
        for (char c = 'A' + rows - i - 1; c >= 'A'; c--) {
            printf("%c ", c);
        }
        printf("\n");
    }
    return 0;
}
```

10. Write a program to generate the multiplication table for a given number as follows

```
Enter the number > 8
+----------------------------------------+
|   8  16  24  32  40  48  56  64  72  80 |
|   1   2   3   4   5   6   7   8   9  10 |
|   8   8   8   8   8   8   8   8   8   8 |
+----------------------------------------+
```

```c
#include <stdio.h>

int main() {
    int num;
    printf("Enter the number > ");
    scanf("%d", &num);
    printf("+----------------------------------------+\n");
    printf("| ");
    for (int i = 1; i <= 10; i++) {// 8 16 24 32....
        printf("%d ", num * i);//Multiple of 8
    }
    printf("|\n");
    printf("| ");
    for (int i = 1; i <= 10; i++) {//1 2 3 4....
        printf("%d ", i); //print numbers
    }
    printf("|\n");
    printf("| ");
    for (int i = 1; i <= 10; i++) {//8 8 8 8 ....
        printf("%d ", num);//print 8
    }
    printf("|\n");
    printf("+----------------------------------------+\n");
    return 0;
}
```