1 Introduction

Decision making is a fundamental concept in programming that allows programs to choose different actions based on conditions. In the C programming language, decision-making structures such as if, if-else, and switch enable developers to control the flow of execution based on specific criteria. These conditional statements evaluate logical expressions and determine the path the program should follow, making them essential tools for handling different scenarios and user inputs. This lab explores the implementation of decision-making structures in C, highlighting their usage and demonstrating how they are applied in practical programming problems.

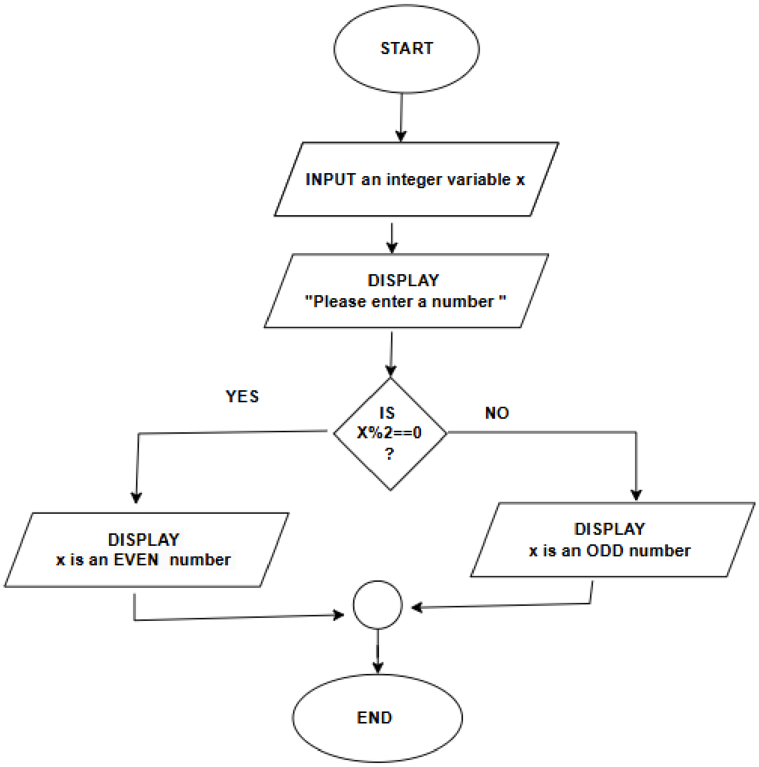
2 C Programs and Flowcharts and Algorithms

2.1 Task: Check Whether a Given Number Is Odd or Even

Algorithm:

```
1. START
2. INPUT $x$
3. DISPLAY ”Please enter a number:”
4. IF $x\%2 == 0$, THEN
     (a) DISPLAY ”$x$ is an EVEN number.”
   ELSE
     (a) DISPLAY”$x$ is an ODD number.”
   END IF
5. END
```

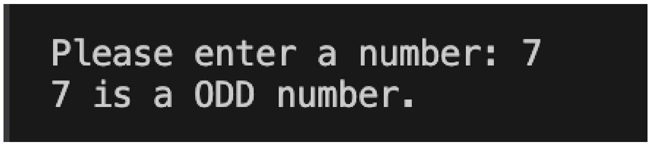
Flowchart:



Code:

```
#include <stdio.h>
int main()
{
    int x;
    printf("Please enter a number: ");
    scanf("%d", &x);

    if (x % 2 == 0)
    {
        printf("%d is an EVEN number.\n", x);
    }
    else
    {
        printf("%d is a ODD number.\n", x);
    }

    return 0;
}
```

Input: 7
Output:



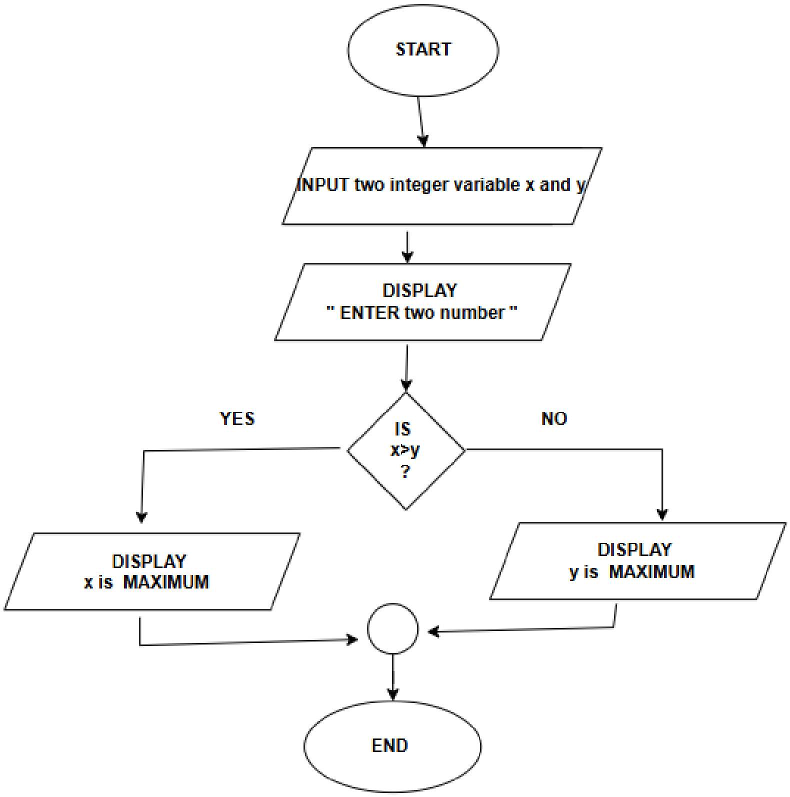
```
Please enter a number: 7
7 is a ODD number.
```

2.2 Task: Find Maximum Between Two Numbers

Algorithm:

```
1. START
2. INPUT $x$ and $y$
3. DISPLAY: "Enter two numbers to compare:"
4. If $x > y$, THEN
      (a) DISPLAY "$x$ is the MAXIMUM."
   ELSE
      (a) DISPLAY "$y$ is the MAXIMUM."
   END IF
5. END
```

Flowchart:

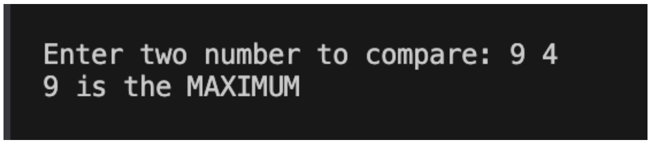


Code:

```
#include <stdio.h>

int main ()
{
    int x,y;
    printf("Enter two number to compare: ");

    scanf("%d %d", &x, &y);

    if (x > y)
    {
        printf("%d is the MAXIMUM\n", x);
    }
    else
    {
       printf("%d is the MAXIMUM\n", y);
    }

    return 0;


}
```

Input: 9 4

Output:



```
Enter two number to compare: 9 4
9 is the MAXIMUM
```

2.3 Task: Check Whether a Number Is Divisible by 5 and 11 or Not

Algorithm:

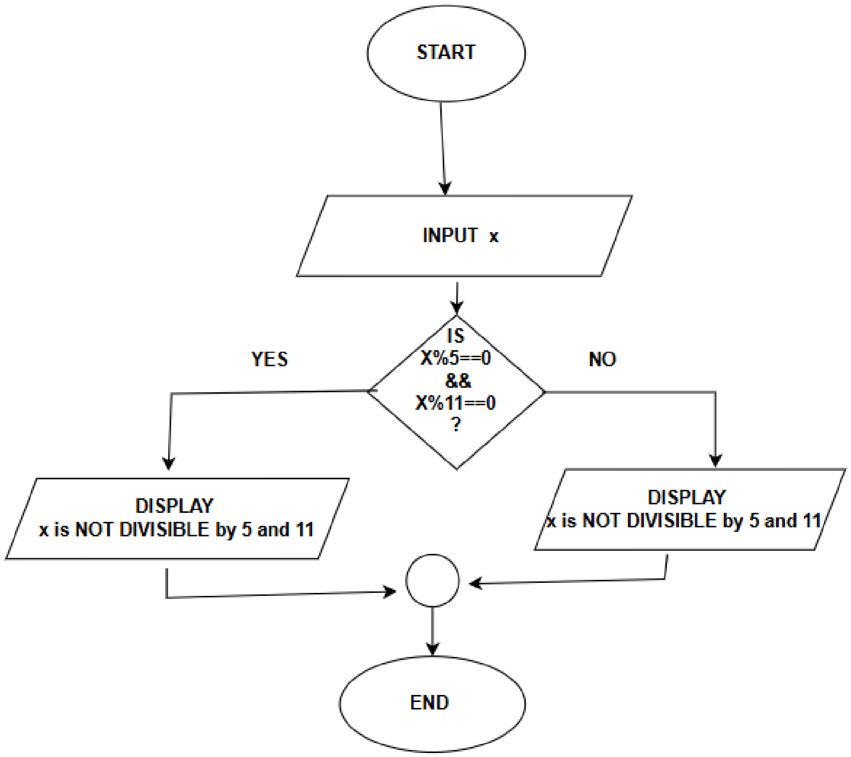
1. START.
2. INPUT x.
3. IF $x\%5 == 0$ AND $x\%11 = 0$ THEN
    - DISPLAY ”x is DIVISIBLE by 5 & 11”.

    ELSE

    - DISPLAY ”x is NOT DIVISIBLE by 5 & 11”.

    END IF.
4. END.

Flowchart:

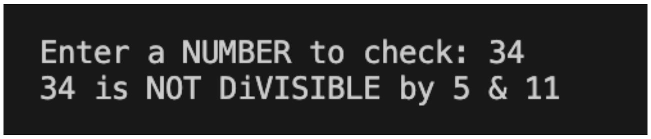

START
INPUT x
IS
X%5==0
&&
X%11==0
?
YES
NO
DISPLAY
x is NOT DIVISIBLE by 5 and 11
DISPLAY
x is NOT DIVISIBLE by 5 and 11
END

Code:

```
#include <stdio.h>

int main()
{
    int x;
    printf("Enter a NUMBER to check: ");
    scanf("%d", &x);

    if (x % 5 == 0 && x % 11 == 0)
    {
    printf("%d is DiVISIBLE by 5 & 11 \n", x);
    }
    else
    {
    printf("%d is NOT DiVISIBLE by 5 & 11 \n", x);
    }

    return 0;

}
```

Input: 34

Output:



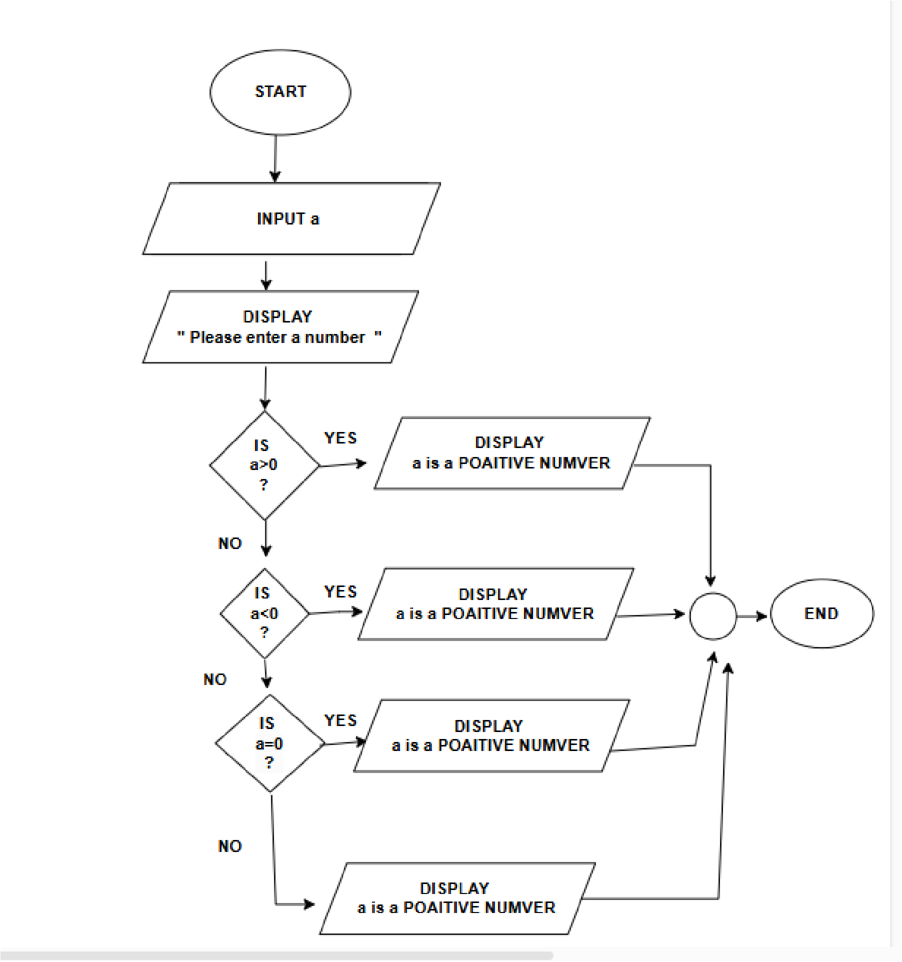
```
Enter a NUMBER to check: 34
34 is NOT DiVISIBLE by 5 & 11
```

2.4 Task: Check Whether a Number Is Negative, Positive, or Zero

Algorithm:

```
1. START.
2. INPUT integer $a$.
3. IF $a > 0$ THEN
       • DISPLAY ”$a$ is a POSITIVE number.”
   ELSE IF $a < 0$ THEN
       • DISPLAY ”$a$ is a NEGATIVE number.”
   ELSE IF $a == 0$ THEN
       • DISPLAY ”Entered number is ZERO.”
   ELSE
       • DISPLAY ”Invalid Number!”
   END IF
4. END.
```

Flowchart:

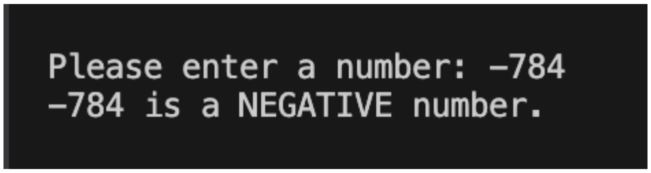


Code:

```
#include <stdio.h>

int main()
{
    int a;
    printf("Please enter a number: ");
    scanf("%d", &a);

    if (a>0)
    {
        printf("%d is a POSITIVE number.\n", a);
    } else if (a < 0)
    {
        printf("%d is a NEGATIVE number.\n", a);
    } else if (a==0)
    {
        printf("Entered number is ZERO.\n");
    } else
    {
        printf("Invalid Number!\n");
    }


    return 0;
}
```

Input: -786

Output:



```
Please enter a number: -784
-784 is a NEGATIVE number.
```

2.5 Task: Check Whether Input Is Vowel or Consonant

Algorithm:

1. START.
2. INPUT *ch*
3. SWITCH *ch*. CASE ’a’:
   - DISPLAY ”*ch* is a VOWEL”.

   CASE ’e’:
   - DISPLAY ”*ch* is a VOWEL”.

   CASE ’i’:
   - DISPLAY ”*ch* is a VOWEL”.

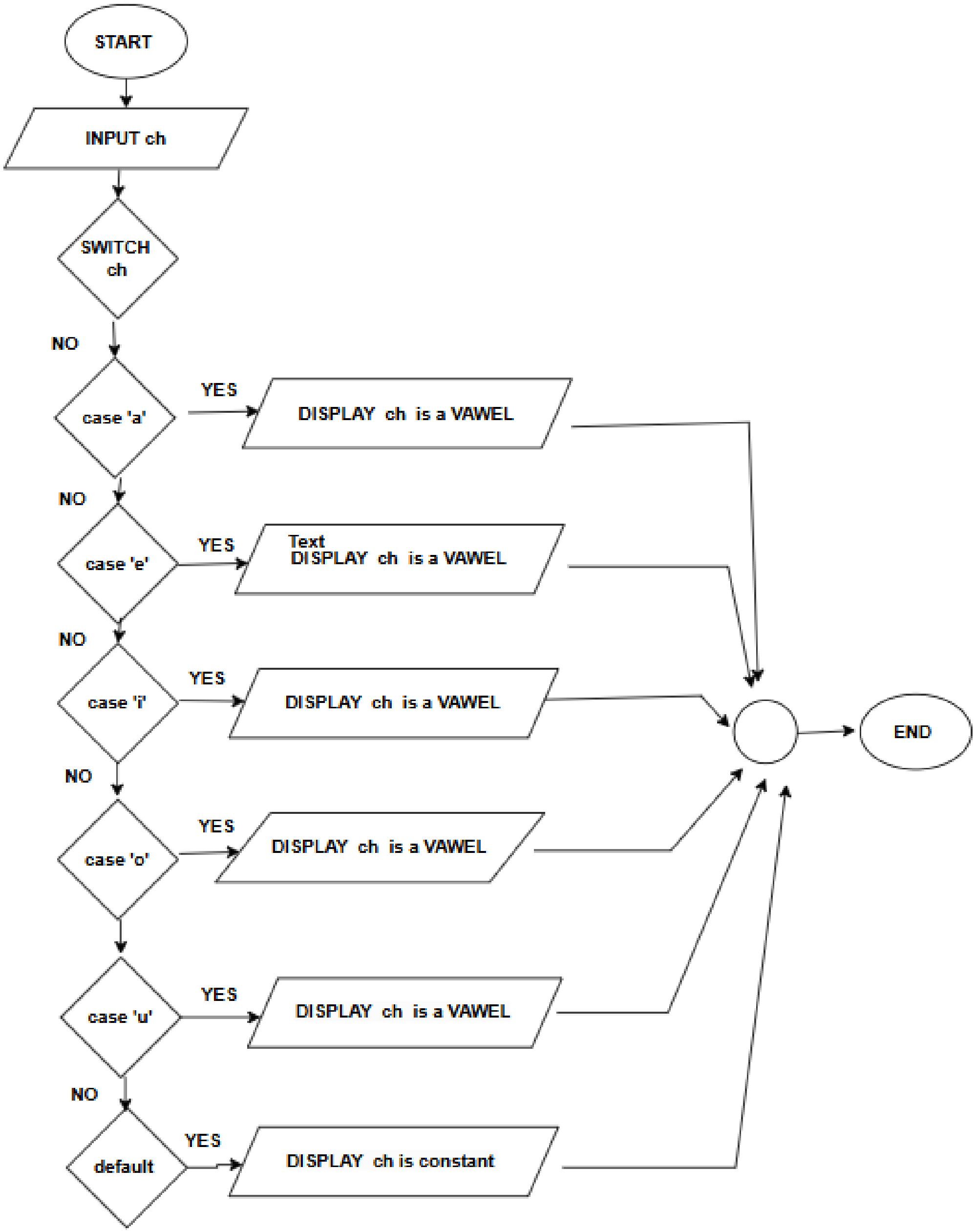
   CASE ’o’:
   - DISPLAY ”*ch* is a VOWEL”.

   CASE ’u’:
   - DISPLAY ”*ch* is a VOWEL”.

   DEFAULT:
   - DISPLAY ”*ch* is a CONSONANT”.

```
    END SWITCH.
4. END.
```

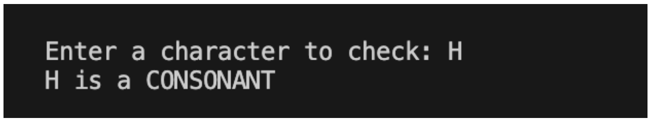
Flowchart:



Code:

```c
#include <stdio.h>

int main()
{
    char ch;
    printf("Enter a character to check: ");
    scanf("%ch", &ch);

    switch (ch)
    {
        case ’a’:
        printf("%c is a VOWEL \n", ch);
        break;

        case ’e’:
        printf("%c is a VOWEL \n", ch);
        break;

        case ’i’:
        printf("%c is a VOWEL \n", ch);
        break;

        case ’o’:
        printf("%c is a VOWEL \n", ch);
        break;

        case ’u’:
        printf("%c is a VOWEL \n", ch);
        break;

        default:
        printf("%c is a CONSONANT \n", ch);
    }

    return 0;
}
```

Input: h
Output:



```
Enter a character to check: H
H is a CONSONANT
```

3 Discussion

In this lab report, we worked on solving problems using if-else and switch-case statements. While the syntax for both is relatively simple, the logic behind them is crucial. One challenge I faced was determining whether an input character is a vowel or a consonant. Using if-else statements for this task would have required more lines of code, so I opted for a switch-case statement to make the code more concise and efficient. This approach helped me understand how to choose the right conditional structure based on the problem's requirements.