PROJECT REPORT:
NUMBER CONVERSION SYSTEM

Group Members:

Student 1: SURYANSH RAGHUWANSHI

Sap id : 590028361

Student 2: VINAY REWAR

Sap id : 590028364

Professor: MR. VINOD KUMAR

Course: Programming in C

1. Problem Definition

Problem Statement: Digital computation relies heavily on the binary (base 2) system, while human communication relies on the decimal (base 10) system. Octal (base 8) and hexadecimal (base 16) are crucial for efficiently representing large binary numbers in computing contexts, such as memory addressing and data representation. Manually converting numbers between these four bases is tedious, time-consuming, and highly susceptible to error, especially with large or complex numerical inputs.

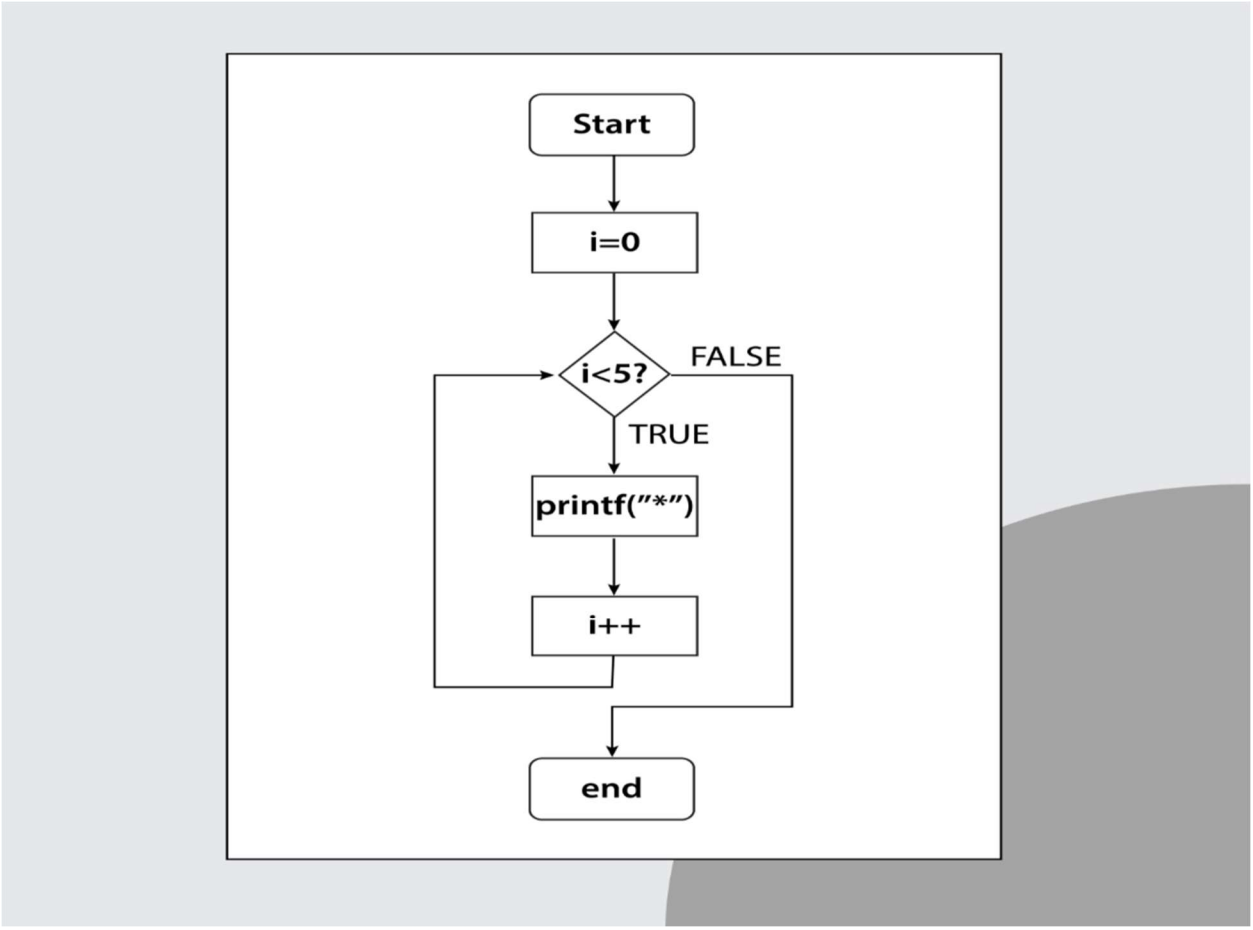
Objective: The goal of this project is to develop an automated system capable of accurately and efficiently performing all necessary conversions between the Decimal, Binary, Octal, and Hexadecimal number systems. The system must validate user inputs and provide accurate output, thereby eliminating manual calculation errors and speeding up development tasks that require base conversion.

2. Flowchart

The following diagram illustrates the fundamental process flow for a universal base conversion tool, focusing on the core method used for converting from Decimal (Base 10) to any target base (Base N, where N=2, 8, \text{or } 16).

Process Flow (Decimal to Base N):

1. **Start.**
2. **Input Decimal Number (D).**
3. **Initialization:** The result (Base N digits) is an empty list.
4. **Loop Condition:** Check if D > 0.
    - If **Yes**: Calculate the Remainder (R = D \bmod N) and the new Quotient (D = \lfloor D / N \rfloor). Add R to the front of the result list.
    - If **No**: Exit the loop.
5. **Output:** Display the concatenated digits (which form the Base N number).
6. **Stop.**



3. Algorithm

The primary algorithm utilized for converting from Decimal to Binary, Octal, or Hexadecimal is the Repeated Division Method.

Algorithm: Decimal to Binary Conversion

This algorithm details the conversion of a number D from Base 10 to Base 2.

Input: Decimal Integer D_{10}. Output: Binary String B_2.

Step	Action	Description
1	Initialize B = \text{empty string}	The binary result is built here.
2	Check Loop Condition: \mathbf{D > 0}	Continues until the decimal number is fully divided.
3	Calculate Remainder: R = D \bmod 2	This remainder is the next binary digit (0 or 1).
4	Update Quotient: D = \lfloor D / 2 \rfloor	Prepares for the next iteration.
5	Concatenate: B = R + B	Prepend the remainder to the result string (MSB is generated last).
6	Repeat Step 2	Loop back until D = 0.
7	Return B	The final binary representation.

Example: Converting 44_{10}:

44 \div 2 = 22, Remainder \mathbf{0}

22 \div 2 = 11, Remainder \mathbf{0}

11 \div 2 = 5, Remainder \mathbf{1}

5 \div 2 = 2, Remainder \mathbf{1}

2 \div 2 = 1, Remainder \mathbf{0}

1 \div 2 = 0, Remainder \mathbf{1}

Reading up the remainders gives 101100_2.

4. Problem Faced by Group

The most significant technical challenge faced during the development of the Number Conversion System was input validation and character-to-value mapping for non-decimal bases, particularly Hexadecimal:

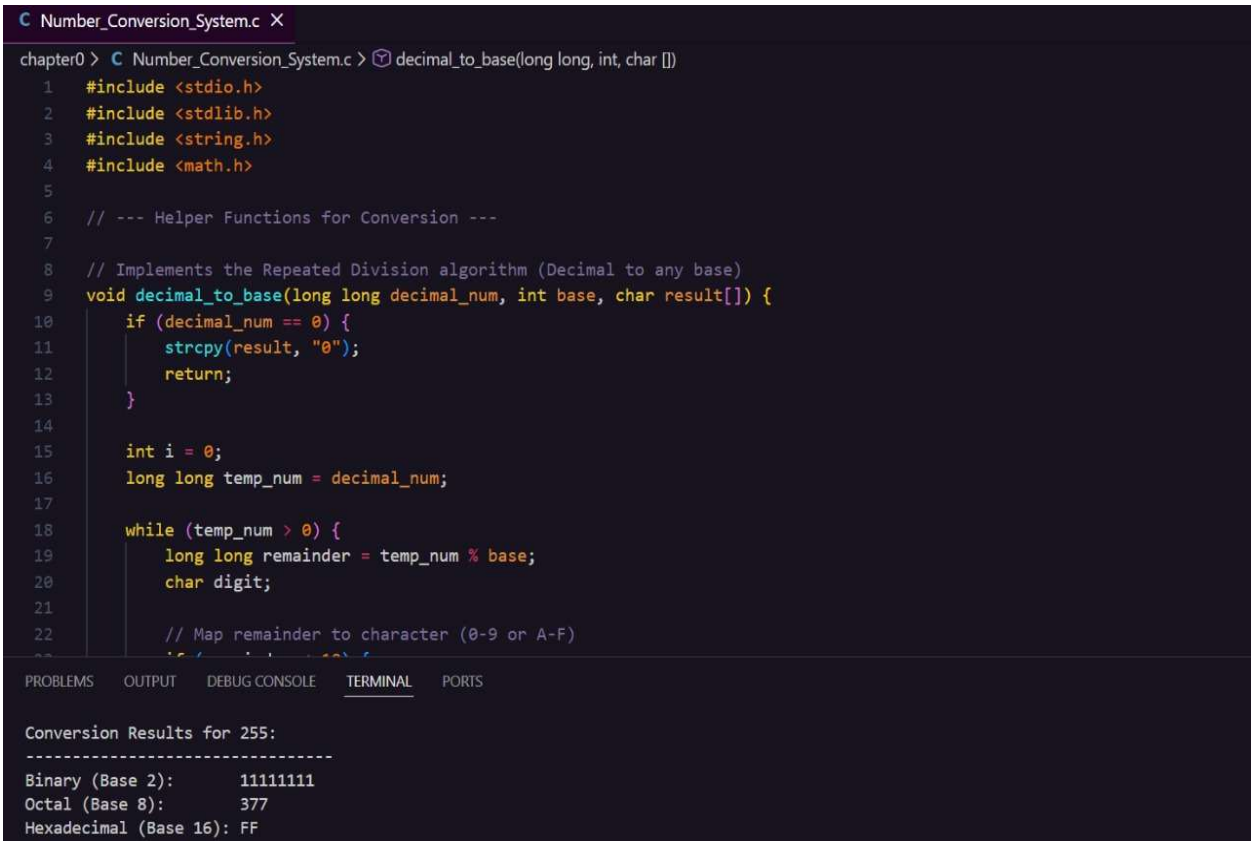
Hexadecimal Input Validation: The system needed to accept characters 'A' through 'F' (or 'a' through 'f') in addition to digits 0-9. This required a robust validation layer to ensure the user did not enter invalid characters (e.g., 'G', 'Z', '!') as part of a Hexadecimal number.

Case Sensitivity: We needed to ensure that 'A' (decimal 10) and 'a' (decimal 10) were treated identically, requiring all inputs to be normalized (e.g., converted to uppercase) before processing.

Octal/Binary Constraints: While simpler, we also had to enforce that Binary inputs contained only '0' or '1', and Octal inputs only contained digits '0' through '7', preventing common errors like entering an '8' in a binary field.

This complexity was resolved by using string manipulation and custom parsing functions to handle the input as text before converting it to an internal numerical representation.

5. Output



The final system offers a clear and intuitive command-line interface. A sample output showing a

three-way conversion from a single Decimal input is demonstrated below:

```
--- Number Conversion System Menu ---
```

1. Convert Decimal (10)
2. Convert Binary (2)
3. Convert Octal (8)
4. Convert Hexadecimal (16)

5. Exit

Enter your choice (1-5): 1

Enter the Decimal Number (D10): 255

Conversion Results for 255:

Binary (Base 2): 11111111
Octal (Base 8): 377
Hexadecimal (Base 16): FF

Press ENTER to return to the menu...

The system ensures that the conversion processes are accurate and nearly instantaneous, providing a reliable utility for handling number base transformations.

6.Final Code

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

// --- Helper Functions for Conversion ---

// Implements the Repeated Division algorithm (Decimal to any base)
void decimal_to_base(long long decimal_num, int base, char result[]) {
if (decimal_num == 0) {
strcpy(result, "0");
return;
}

int i = 0;
long long temp_num = decimal_num;

while (temp_num > 0) {
long long remainder = temp_num % base;
char digit;

// Map remainder to character (0-9 or A-F)
if (remainder < 10) {
digit = remainder + '0';
} else {
digit = remainder - 10 + 'A';
}

// Store digit (in reverse order)
result[i++] = digit;
temp_num /= base;
}
result[i] = '\0';

// Reverse the string to get the correct order (MSB first)
int len = strlen(result);
for (int j = 0; j < len / 2; j++) {
char temp = result[j];
result[j] = result[len - 1 - j];
result[len - 1 - j] = temp;
}
}

// Converts any base (Binary, Octal, Hex) string to Decimal
long long base_to_decimal(const char *num_str, int base) {
long long decimal_num = 0;
int len = strlen(num_str);
int power = 0;
```

```
// Iterate from the rightmost digit (least significant bit/digit)
for (int i = len - 1; i >= 0; i--) {
char digit = num_str[i];
int value;

if (digit >= '0' && digit <= '9') {
value = digit - '0';
} else if (digit >= 'A' && digit <= 'F') {
value = digit - 'A' + 10;
} else if (digit >= 'a' && digit <= 'f') {
// Handle lowercase hex input
value = digit - 'a' + 10;
} else {
// Invalid character for the base
return -1;
}

// Validate the digit's value against the base
if (value >= base) {
return -2; // Conversion error: digit exceeds base
}

decimal_num += value * (long long)pow(base, power);
power++;
}
return decimal_num;
}

// --- Main Program Functions ---

void display_menu() {
printf("\n--- Number Conversion System Menu ---\n");
printf("1. Decimal (10) to Other Bases\n");
printf("2. Binary (2) to Decimal\n");
printf("3. Octal (8) to Decimal\n");
printf("4. Hexadecimal (16) to Decimal\n");
printf("5. Exit\n");
printf("-------------------------------------\n");
}

void decimal_to_all() {
long long dec;
printf("Enter Decimal Number (Base 10): ");
if (scanf("%lld", &dec) != 1 || dec < 0) {
printf("Invalid input. Please enter a positive integer.\n");
// Clear input buffer
while (getchar() != '\n');
return;
}
```

```
char binary_res[64];
char octal_res[64];
char hex_res[64];

decimal_to_base(dec, 2, binary_res);
decimal_to_base(dec, 8, octal_res);
decimal_to_base(dec, 16, hex_res);

printf("\nConversion Results for %lld:\n", dec);
printf("---------------------------------\n");
printf("Binary (Base 2):       %s\n", binary_res);
printf("Octal (Base 8):        %s\n", octal_res);
printf("Hexadecimal (Base 16): %s\n", hex_res);
printf("---------------------------------\n");
}

void any_base_to_decimal(int base) {
char input_str[64];
printf("Enter Number in Base %d: ", base);
scanf("%63s", input_str);

// Convert input to uppercase for uniform processing (especially for Hex)
for (int i = 0; input_str[i]; i++) {
if (input_str[i] >= 'a' && input_str[i] <= 'z') {
input_str[i] = input_str[i] - 32;
}
}

long long dec = base_to_decimal(input_str, base);

printf("---------------------------------\n");
if (dec == -1) {
printf("ERROR: Invalid character detected in the input for Base %d.\n", base);
} else if (dec == -2) {
printf("ERROR: Invalid digit detected (e.g., 8 or 9 in Binary/Octal).\n");
} else {
printf("Input (%s in Base %d) = Decimal (Base 10): %lld\n", input_str, base, dec);
}
printf("---------------------------------\n");
}


int main() {
int choice;

// Log level for debugging (not strictly needed for this C code, but good practice)
// setLogLevel("Debug");

do {
display_menu();
printf("Enter your choice (1-5): ");
```

```
// Input validation for menu choice
if (scanf("%d", &choice) != 1) {
printf("\nERROR: Invalid input. Please enter a number.\n");
// Clear input buffer on error
while (getchar() != '\n');
choice = 0; // Set choice to an invalid value to continue loop
} else {
// Clear buffer after successful read
while (getchar() != '\n');
}

switch (choice) {
case 1:
decimal_to_all();
break;
case 2:
any_base_to_decimal(2);
break;
case 3:
any_base_to_decimal(8);
break;
case 4:
any_base_to_decimal(16);
break;
case 5:
printf("\nExiting Number Conversion System. Goodbye!\n");
break;
default:
if (choice != 0) {
printf("\nInvalid option. Please choose a number between 1 and 5.\n");
}
break;
}
} while (choice != 5);

return 0;
}
```

END