Batch: E-2	**Roll No.: 16010123325**
Experiment / assignment / tutorial No. 5	

TITLE: Implementation of IEEE-754 floating point representation

AIM: To demonstrate the single and double precision formats to represent floating point numbers.

Expected OUTCOME of Experiment: (Mention CO attained here)

Books/ Journals/ Websites referred:

1. Carl Hamacher, Zvonko Vranesic and Safwat Zaky, "Computer Organization", Fifth Edition, TataMcGraw-Hill.
2. William Stallings, "Computer Organization and Architecture: Designing for Performance", Eighth Edition, Pearson.

Pre Lab/ Prior Concepts:

The IEEE Standard for Floating-Point Arithmetic (IEEE 754) is a technical standard for floating-point computation established in 1985 by the Institute of Electrical and Electronics Engineers (IEEE). The standard addressed many problems found in the diverse floating point implementations that made them difficult to use reliablyand portably. Many hardware floating point units now use the IEEE 754 standard.

The standard defines:

- *arithmetic formats:* sets of binary and decimal floating-point data, which consist of finite numbers (including signed zeros and subnormal numbers), infinities, and special "not a number" values (NaNs)
- *interchange formats:* encodings (bit strings) that may be used to exchange floating-point data in an efficient and compact form
- *rounding rules:* properties to be satisfied when rounding numbers during arithmetic and conversions
- *operations:* arithmetic and other operations (such as trigonometric functions) on arithmetic formats
- *exception handling:* indications of exceptional conditions (such as division by zero, overflow, *etc*

Code-

```
#include <bits/stdc++.h>
#include <bitset>
using namespace std;
#define uint unsigned int
#define ull unsigned long long

typedef union {
    float f;
    struct {
        uint mantissa: 23;
        uint exponent: 8;
        uint sign: 1;
    } raw;
} myfloat32;

typedef union {
    double f;
    struct {
        ull mantissa: 52;
        uint exponent: 11;
        uint sign: 1;
    } raw;
} myfloat64;

void printBinary(int n) {
    bitset<32> b(n);
    cout << b.to_string() << '\n';
}

void printBinary64(ull n) {
    bitset<64> b(n);
    cout << b.to_string() << '\n';
}

void printIEEE32(myfloat32 var) {
    cout << "Sign: " << var.raw.sign << '\n';
    cout << "Exponent: ";
    printBinary(var.raw.exponent);
    cout << "Mantissa: ";
    printBinary(var.raw.mantissa);
    cout << '\n';
}
```

```
void printIEEE64(myfloat64 var) {
    cout << "Sign: " << var.raw.sign << '\n';
    cout << "Exponent: ";
    printBinary(var.raw.exponent);
    cout << "Mantissa: ";
    printBinary64(var.raw.mantissa);
    cout << '\n';
}

int main() {
    int choice;
    cout << "Enter 1 for 32-bit IEEE 754 or 2 for 64-bit IEEE 754: ";
    cin >> choice;
    if (choice == 1) {
        myfloat32 var;
        cout << "Enter a decimal number: ";
        cin >> var.f;
        printIEEE32(var);
    } else if (choice == 2) {
        myfloat64 var;
        cout << "Enter a decimal number: ";
        cin >> var.f;
        printIEEE64(var);
    } else {
        cout << "Invalid choice.\n";
    }

    return 0;
}
```

Output-

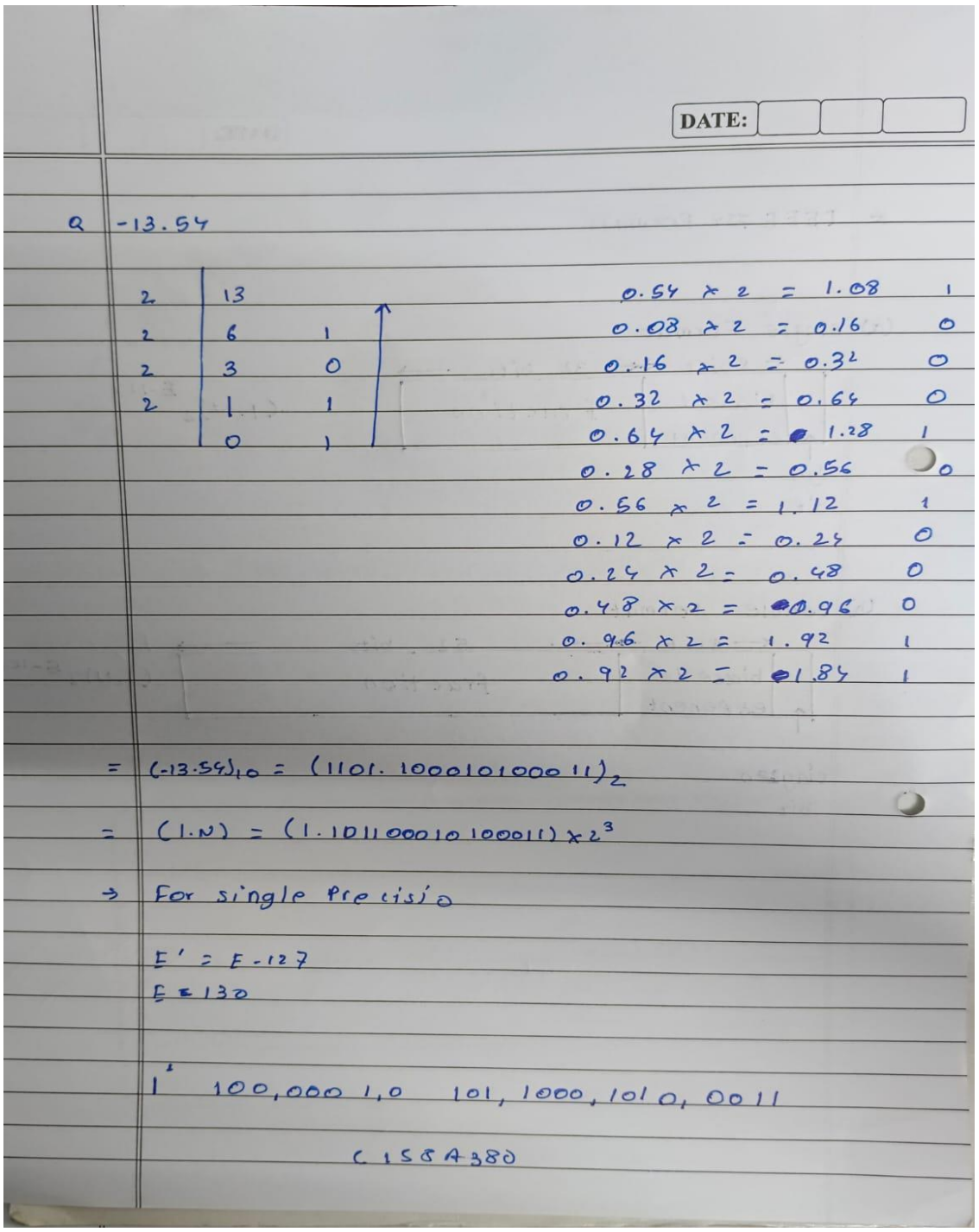
```
PS C:\Users\Shrey\OneDrive\Desktop\KJSCE\SEM-3\COA\Programs> cd "c:\Users\Shrey\OneDrive\Desktop\KJSCE\SEM-3\COA\Programs\" ; i
f ($?) { g++ IEEEE-floating-point.cpp -o IEEEE-floating-point } ; if ($?) { .\IEEEE-floating-point }
Enter 1 for 32-bit IEEE 754 or 2 for 64-bit IEEE 754: 1
Enter a decimal number: 125.25
Sign: 0
Exponent: 00000000000000000000000010000101
Mantissa: 00000000011110101000000000000000

PS C:\Users\Shrey\OneDrive\Desktop\KJSCE\SEM-3\COA\Programs>
```

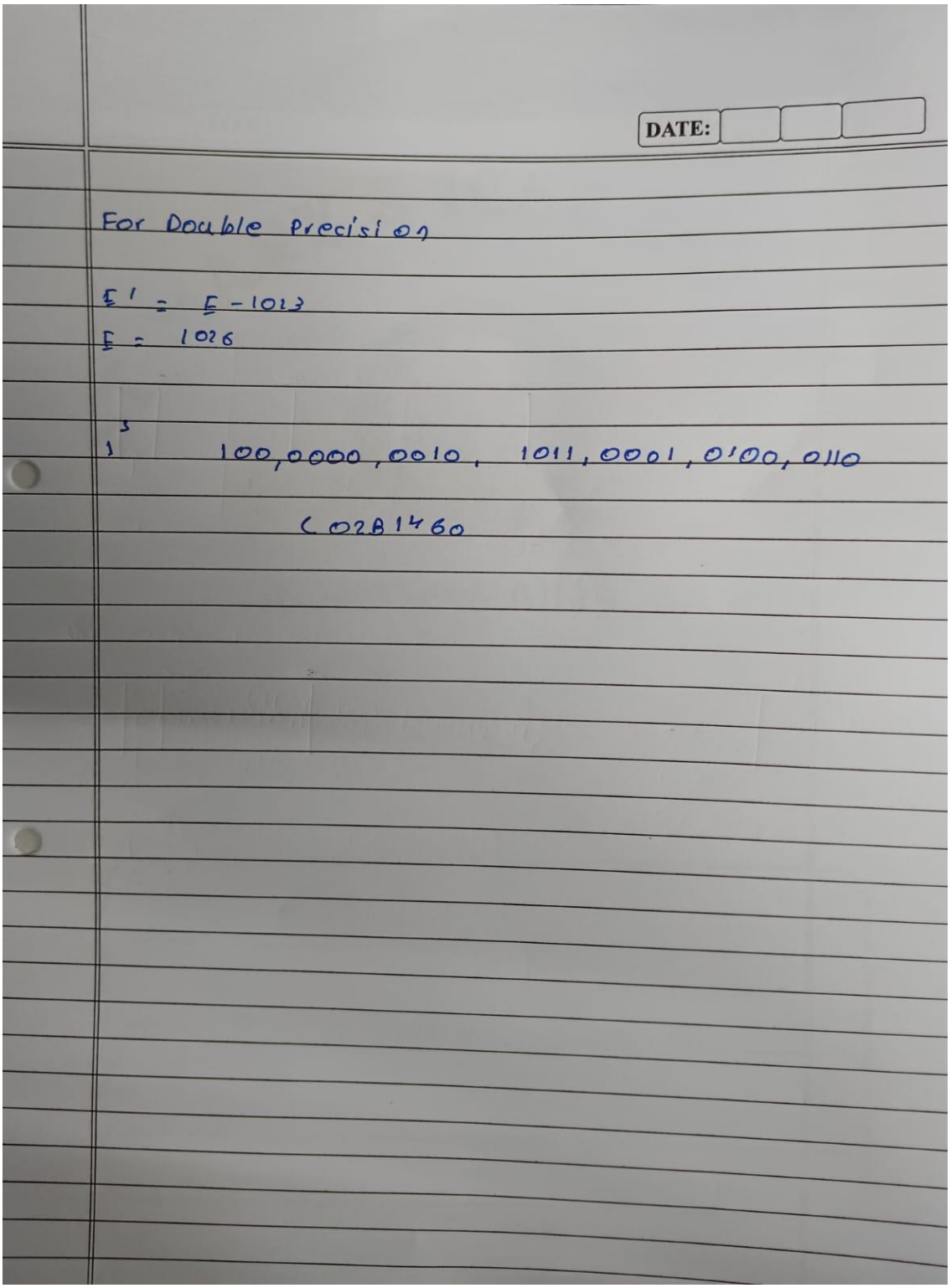
```
PS C:\Users\Shrey\OneDrive\Desktop\KJSCE\SEM-3\COA\Programs> cd "c:\Users\Shrey\OneDrive\Desktop\KJSCE\SEM-3\COA\Programs\" ; i
f ($?) { g++ IEEEE-floating-point.cpp -o IEEEE-floating-point } ; if ($?) { .\IEEEE-floating-point }
Enter 1 for 32-bit IEEE 754 or 2 for 64-bit IEEE 754: 2
Enter a decimal number: 125.25
Sign: 0
Exponent: 00000000000000000000001001101101
Mantissa: 00000000000011110101000000000000000000000000000000000000000000000

PS C:\Users\Shrey\OneDrive\Desktop\KJSCE\SEM-3\COA\Programs>
```

```
PS C:\Users\Shrey\OneDrive\Desktop\KJSCE\SEM-3\COA\Programs> cd "c:\Users\Shrey\OneDrive\Desktop\KJSCE\SEM-3\COA\Programs\" ;
f ($?) { g++ IEEEE-floating-point.cpp -o IEEEE-floating-point } ; if ($?) { .\IEEEE-floating-point }
Enter 1 for 32-bit IEEE 754 or 2 for 64-bit IEEE 754: 1
Enter a decimal number: 0.0635
Sign: 0
Exponent: 00000000000000000000000001111011
Mantissa: 00000000000000100000110001001010

PS C:\Users\Shrey\OneDrive\Desktop\KJSCE\SEM-3\COA\Programs>
```

```
PS C:\Users\Shrey\OneDrive\Desktop\KJSCE\SEM-3\COA\Programs> cd "c:\Users\Shrey\OneDrive\Desktop\KJSCE\SEM-3\COA\Programs\" ; i
f ($?) { g++ IEEEE-floating-point.cpp -o IEEEE-floating-point } ; if ($?) { .\IEEEE-floating-point }
Enter 1 for 32-bit IEEE 754 or 2 for 64-bit IEEE 754: 2
Enter a decimal number: 0.0635
Sign: 0
Exponent: 00000000000000000000001001101101
Mantissa: 0000000000000000100000110001001001101110100101111000110101010000

PS C:\Users\Shrey\OneDrive\Desktop\KJSCE\SEM-3\COA\Programs>
```

Example (Single Precision- 32 bit representation)/ (Double Precision- 64 bit representation)




DATE:
For Double Precision
E' = E - 1023
E = 1026
1 5
1 100,0000,0010, 1011,0001,0100,0110
C02B1460

Post Lab Descriptive Questions

Give the importance of IEEE-754 representation for floating point numbers?

The **IEEE-754** standard is crucial for representing floating-point numbers consistently across various computing systems. It defines how real numbers are stored using a combination of sign, exponent, and mantissa, allowing for both very large and very small numbers to be represented with a degree of precision. This standard also defines special cases like **NaN** (Not a Number), **Infinity**, and **subnormal numbers**, which helphandle arithmetic errors, overflow, and underflow gracefully. The use of IEEE-754 ensures portability, meaning floating-point calculations give the same results on different hardware and software platforms.

Conclusion

Hence, the above experiment highlights the importance of IEEE-754 representation of floating point numbers and their implementation in code using Java.

Date: _____