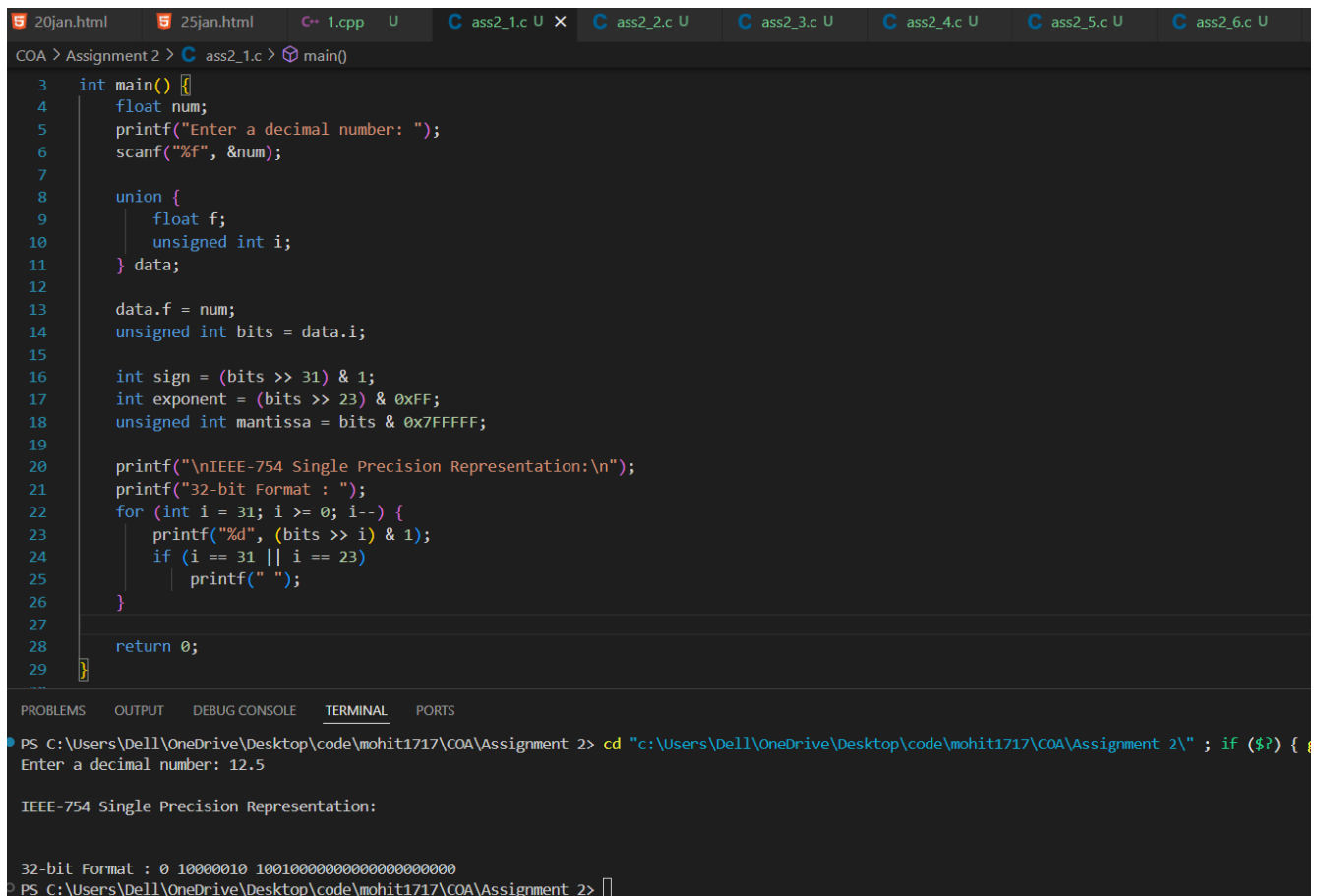


Q1



The image shows a C++ IDE with a dark theme. The top toolbar includes icons for file operations and tabs for various files: 20jan.html, 25jan.html, 1.cpp, and several ass2\_\*.c files. The active file is 'ass2\_1.c'. The editor displays a C++ program that takes a decimal number and outputs its IEEE-754 single precision representation in 32-bit format. The code uses a union to access the raw bits of a float. The terminal window at the bottom shows the command prompt, the execution of the program, the input '12.5', and the resulting 32-bit binary representation: 0 1000010 100100000000000000000000.

```
3 int main() {
4     float num;
5     printf("Enter a decimal number: ");
6     scanf("%f", &num);
7
8     union {
9         float f;
10        unsigned int i;
11    } data;
12
13    data.f = num;
14    unsigned int bits = data.i;
15
16    int sign = (bits >> 31) & 1;
17    int exponent = (bits >> 23) & 0xFF;
18    unsigned int mantissa = bits & 0x7FFFFFFF;
19
20    printf("\nIEEE-754 Single Precision Representation:\n");
21    printf("32-bit Format : ");
22    for (int i = 31; i >= 0; i--) {
23        printf("%d", (bits >> i) & 1);
24        if (i == 31 || i == 23)
25            printf(" ");
26    }
27
28    return 0;
29 }
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

PS C:\Users\Dell\OneDrive\Desktop\code\mohit1717\COA\Assignment 2> cd "c:\Users\Dell\OneDrive\Desktop\code\mohit1717\COA\Assignment 2\" ; if (\$?) { g++ ass2\_1.c -std=c++11 -o ass2\_1.exe ; .\ass2\_1.exe }

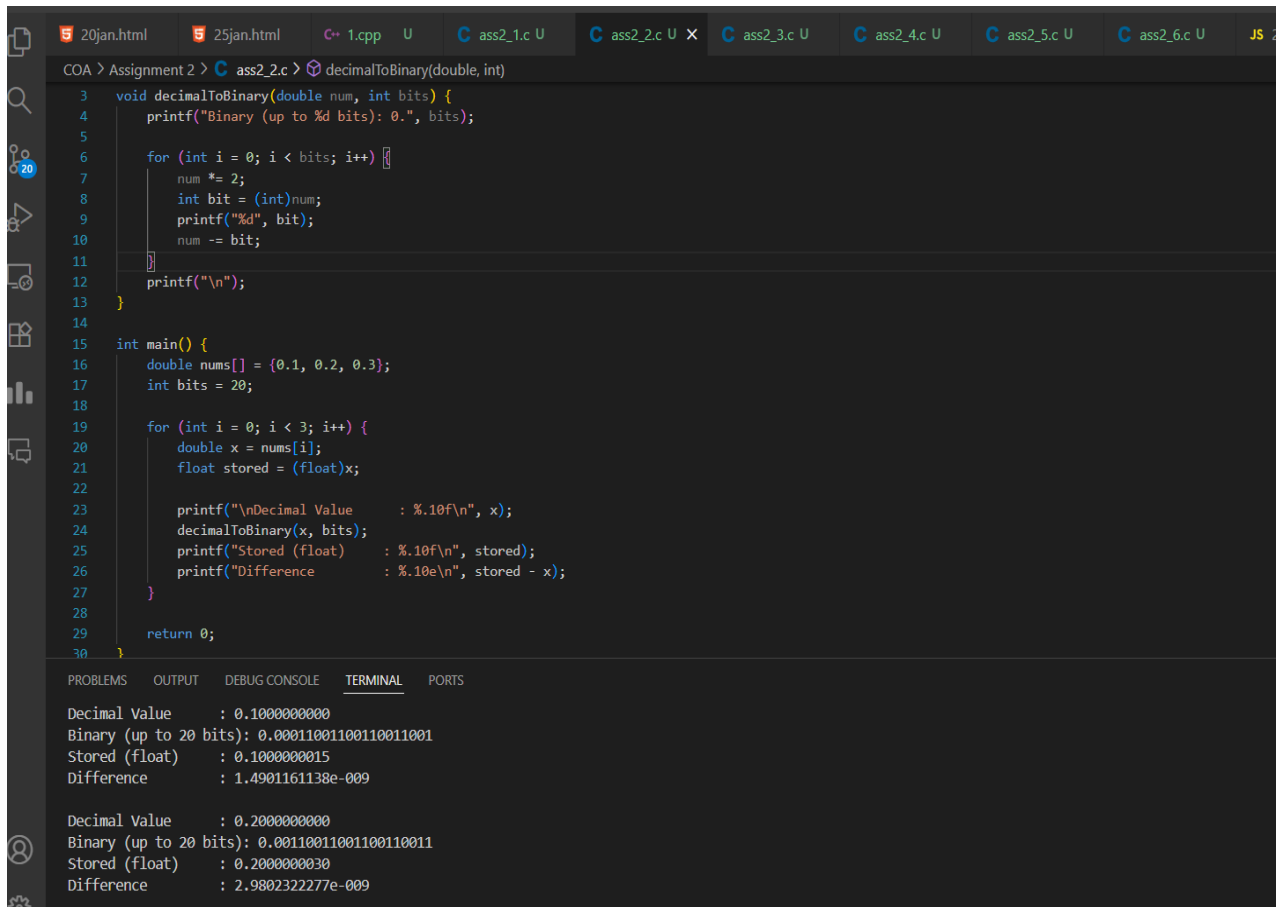
Enter a decimal number: 12.5

IEEE-754 Single Precision Representation:

32-bit Format : 0 1000010 100100000000000000000000

PS C:\Users\Dell\OneDrive\Desktop\code\mohit1717\COA\Assignment 2> |

Q 2



The screenshot shows a C++ IDE with a project named 'COA > Assignment 2'. The active file is 'ass2\_2.c', which contains a function 'decimalToBinary' and a 'main' function. The 'decimalToBinary' function takes a double 'num' and an integer 'bits' as input. It prints the binary representation of 'num' up to 'bits' by repeatedly multiplying by 2 and extracting the integer part. The 'main' function tests this with three values: 0.1, 0.2, and 0.3. For each value, it prints the decimal value, its binary representation (up to 20 bits), the value stored as a float, and the difference between the original decimal value and the stored float value.

```
3 void decimalToBinary(double num, int bits) {
4     printf("Binary (up to %d bits): 0.", bits);
5
6     for (int i = 0; i < bits; i++) {
7         num *= 2;
8         int bit = (int)num;
9         printf("%d", bit);
10        num -= bit;
11    }
12    printf("\n");
13 }
14
15 int main() {
16     double nums[] = {0.1, 0.2, 0.3};
17     int bits = 20;
18
19     for (int i = 0; i < 3; i++) {
20         double x = nums[i];
21         float stored = (float)x;
22
23         printf("\nDecimal Value      : %.10f\n", x);
24         decimalToBinary(x, bits);
25         printf("Stored (float)       : %.10f\n", stored);
26         printf("Difference          : %.10e\n", stored - x);
27     }
28
29     return 0;
30 }
```

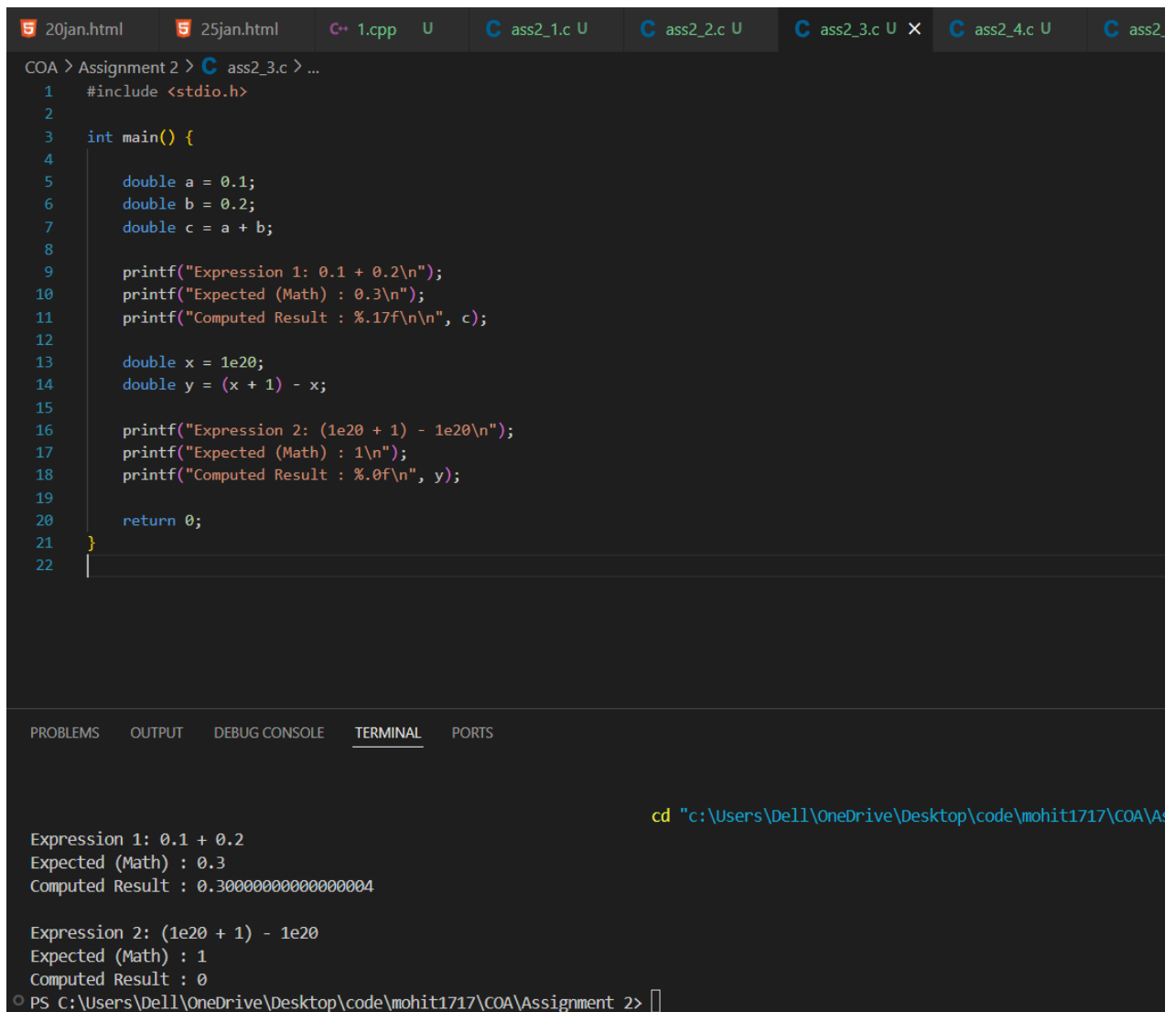
The terminal output shows the results for the three test cases:

```
Decimal Value      : 0.1000000000
Binary (up to 20 bits): 0.00011001100110011001
Stored (float)     : 0.1000000015
Difference         : 1.4901161138e-009

Decimal Value      : 0.2000000000
Binary (up to 20 bits): 0.00110011001100110011
Stored (float)     : 0.2000000030
Difference         : 2.9802322277e-009

Decimal Value      : 0.3000000000
Binary (up to 20 bits): 0.01001100110011001100
Stored (float)     : 0.3000000000
Difference         : 0.0000000000e+000
```

Q3



The screenshot shows a C++ IDE with a file explorer at the top displaying several files: 20jan.html, 25jan.html, 1.cpp, ass2\_1.c, ass2\_2.c, ass2\_3.c, ass2\_4.c, and ass2\_5.c. The main editor window shows the code for 'ass2\_3.c'. The code defines two expressions and compares their computed results with expected mathematical results. The first expression is  $0.1 + 0.2$ , and the second is  $(1e20 + 1) - 1e20$ . The output window shows the results of these calculations, highlighting a significant floating-point precision error in the first case.

```
COA > Assignment 2 > C ass2_3.c > ...
1  #include <stdio.h>
2
3  int main() {
4
5      double a = 0.1;
6      double b = 0.2;
7      double c = a + b;
8
9      printf("Expression 1: 0.1 + 0.2\n");
10     printf("Expected (Math) : 0.3\n");
11     printf("Computed Result : %.17f\n", c);
12
13     double x = 1e20;
14     double y = (x + 1) - x;
15
16     printf("Expression 2: (1e20 + 1) - 1e20\n");
17     printf("Expected (Math) : 1\n");
18     printf("Computed Result : %.0f\n", y);
19
20     return 0;
21 }
22
```

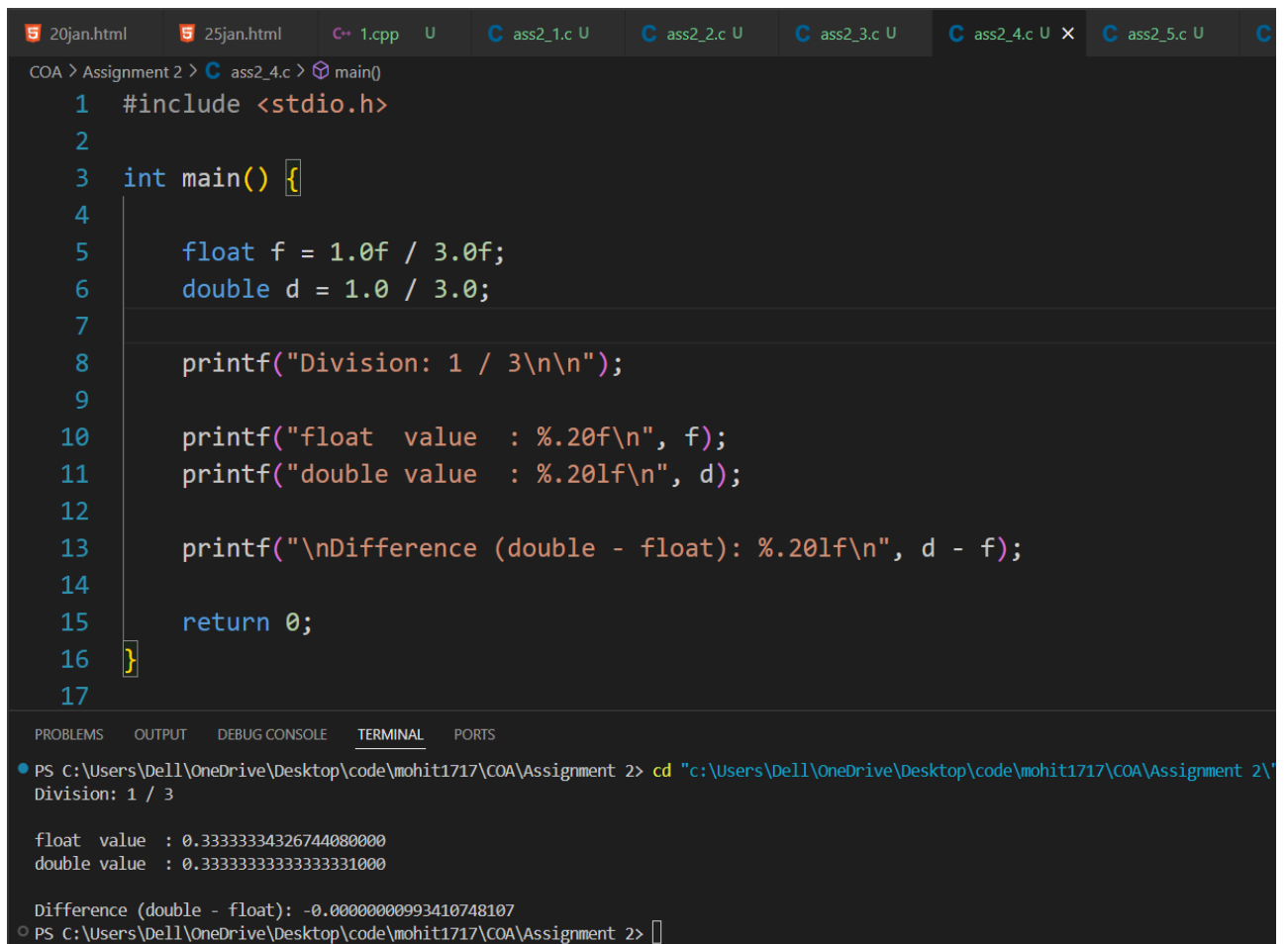
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

```
cd "c:\Users\Dell\OneDrive\Desktop\code\mohit1717\COA\As

Expression 1: 0.1 + 0.2
Expected (Math) : 0.3
Computed Result : 0.30000000000000004

Expression 2: (1e20 + 1) - 1e20
Expected (Math) : 1
Computed Result : 0
PS C:\Users\Dell\OneDrive\Desktop\code\mohit1717\COA\Assignment 2>
```

Q4



The image shows a Visual Studio Code editor window with a C program being edited and run. The editor has several tabs at the top: 20jan.html, 25jan.html, 1.cpp, and several ass2\_\*.c files. The active file is ass2\_4.c, and the cursor is at the end of the main() function. The code in the editor is as follows:

```
1 #include <stdio.h>
2
3 int main() {
4
5     float f = 1.0f / 3.0f;
6     double d = 1.0 / 3.0;
7
8     printf("Division: 1 / 3\n\n");
9
10    printf("float value : %.20f\n", f);
11    printf("double value : %.20lf\n", d);
12
13    printf("\nDifference (double - float): %.20lf\n", d - f);
14
15    return 0;
16 }
17
```

Below the editor, the TERMINAL tab is active, showing the output of the program. The command prompt shows the current directory and the command to run the program. The output is as follows:

```
PS C:\Users\Dell\OneDrive\Desktop\code\mohit1717\COA\Assignment 2> cd "c:\Users\Dell\OneDrive\Desktop\code\mohit1717\COA\Assignment 2\"
Division: 1 / 3

float value : 0.33333334326744080000
double value : 0.33333333333333331000

Difference (double - float): -0.00000000993410748107
PS C:\Users\Dell\OneDrive\Desktop\code\mohit1717\COA\Assignment 2>
```

Q 5



```
20jan.html 25jan.html 1.cpp U ass2_1.c U ass2_2.c U ass2_3.c U ass2_4.c U ass2_5.c U X
COA > Assignment 2 > C ass2_5.c > main()
1  #include <stdio.h>
2  #include <float.h>
3
4  int main()
5  {
6      float x = FLT_MAX;
7      float y = FLT_MIN;
8
9      printf("Overflow example:\n");
10     printf("FLT_MAX = %e\n", x);
11
12     x = x * 10;
13     printf("After overflow = %e\n", x);
14
15     printf("\nUnderflow example:\n");
16     printf("FLT_MIN = %e\n", y);
17
18     y = y / 10;
19     printf("After underflow = %e\n", y);
20
21     return 0;
22 }
23
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

```
cd "c:\Users\Dell\OneDrive\Desktop\code\mohit1717\COA\Assignment 2"
Overflow example:
FLT_MAX = 3.402823e+038
After overflow = 1.#INF00e+000

Underflow example:
FLT_MIN = 1.175494e-038
After underflow = 1.175495e-039
PS C:\Users\Dell\OneDrive\Desktop\code\mohit1717\COA\Assignment 2>
```


Q 6

COA > Assignment 2 >  ass2\_6.c >  main()

```
1  #include<stdio.h>
2  #include<float.h>
3
4  int main(){
5
6      float num= FLT_MIN;
7
8      printf("Normalized minimum values : %e\n",num);
9
10     num=num/2;
11     printf("Subnormal value : %e\n",num);
12
13
14     return 0;
15 }
```

PROBLEMS   OUTPUT   DEBUG CONSOLE   TERMINAL   PORTS

Normalized minimum values : 1.175494e-038  
Subnormal value : 5.877472e-039

 "c:\Users\Dell\OneDrive\Desktop\code\mohit1717\COA\A