18th September 2024

Lab 1

CSCI291

University of Wollongong in Dubai

School of Engineering

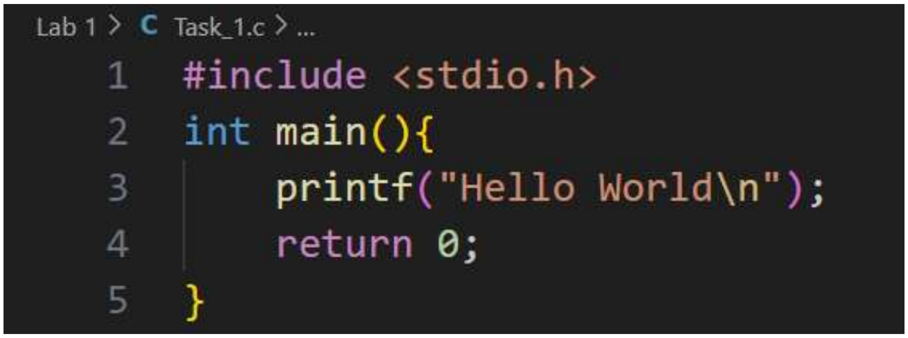
CSCI291

Dana Hasan Alhafidh

8215765

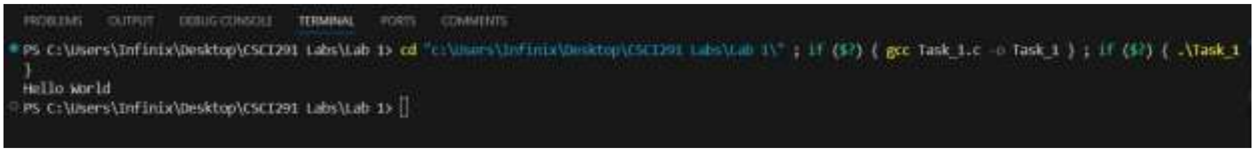
Task 1: Hello World program

Input:



```c
#include <stdio.h>
int main(){
    printf("Hello World\n");
    return 0;
}
```

Output:



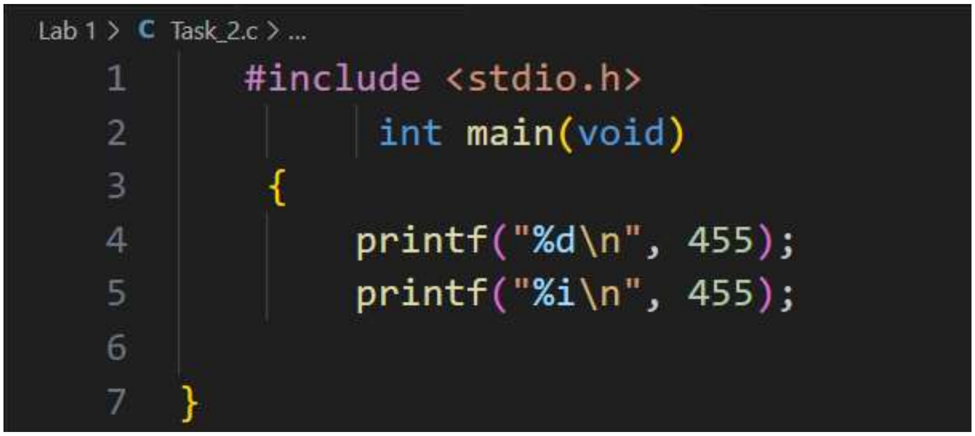
```
PS C:\Users\Infinix\Desktop\CSCI291 Labs\Lab 1> cd "c:\Users\Infinix\Desktop\CSCI291 Labs\Lab 1\" ; if ($?) { gcc Task_1.c -o Task_1 } ; if ($?) { .\Task_1
}
Hello World
PS C:\Users\Infinix\Desktop\CSCI291 Labs\Lab 1> 
```

Task 2: printf statements

%d & %i:

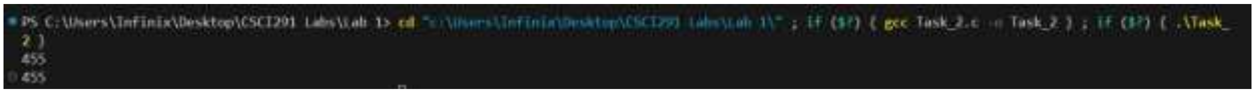
1- Both yield the same results/outcome especially when printf is used. d and i mean integer

Input:



```c
    #include <stdio.h>
            int main(void)
     {
            printf("%d\n", 455);
            printf("%i\n", 455);

 }
```

Output:

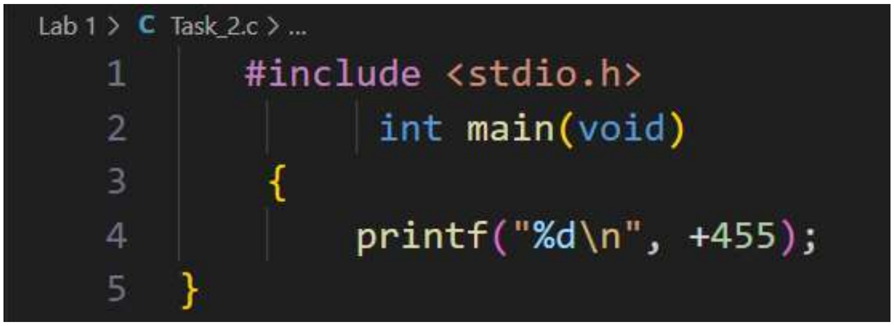


```
PS C:\Users\Infinix\Desktop\CSCI291 Labs\Lab 1> cd "c:\Users\Infinix\Desktop\CSCI291 Labs\Lab 1\" ; if ($?) { gcc Task_2.c -o Task_2 } ; if ($?) { .\Task_
2 }
455
455
```

<u>%d Positive integer</u>

2- %d is an integer in this case a positive integer

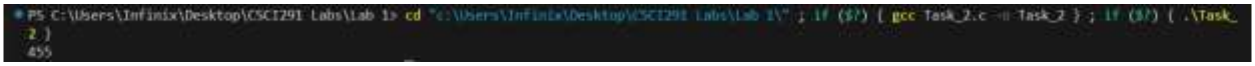
Input:



Lab 1 > Task_2.c > ...

```
    #include <stdio.h>
            int main(void)
      {
            printf("%d\n", +455);
}
```

Output:

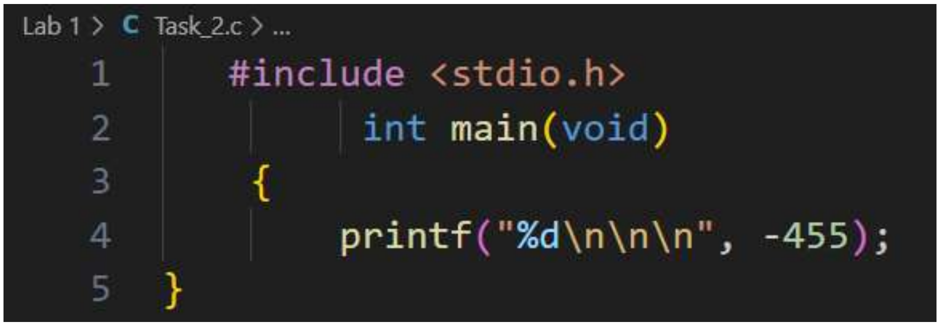


```
PS C:\Users\Infinix\Desktop\CSCI291 Labs\Lab 1> cd "c:\Users\Infinix\Desktop\CSCI291 Labs\Lab 1\" ; if ($?) { gcc Task_2.c -o Task_2 } ; if ($?) { .\Task_
2 }
455
```

<u>%d, a negative value</u>

3- In this case, %d is an integer and negative. \n means new line so \n\n\n means 3 new lines

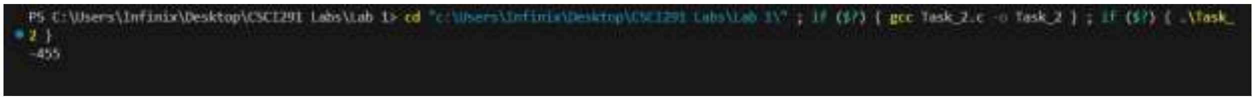
Input:



Lab 1 > Task_2.c > ...

```
    #include <stdio.h>
            int main(void)
      {
            printf("%d\n\n\n", -455);
}
```

Output:

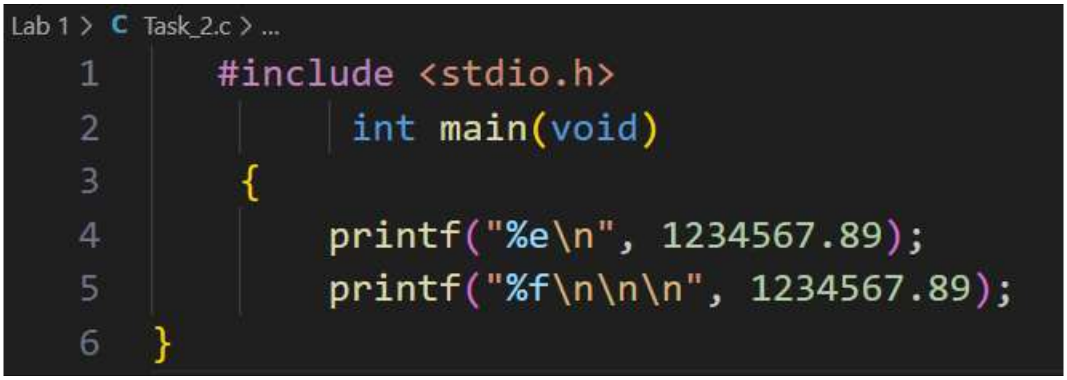


```
PS C:\Users\Infinix\Desktop\CSCI291 Labs\Lab 1> cd "c:\Users\Infinix\Desktop\CSCI291 Labs\Lab 1\" ; if ($?) { gcc Task_2.c -o Task_2 } ; if ($?) { .\Task_
2 }
-455
```

<u>%d & %f</u>

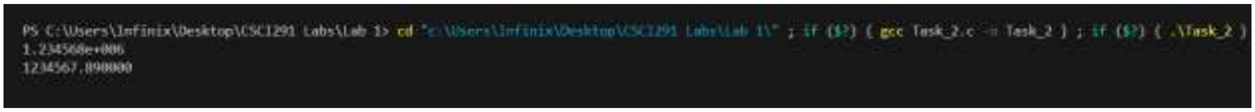
4- %e is a floating-point number in exponential format (a scientific notation). %f is a floating-point number, to six decimal places by default.

Input:



```c
    #include <stdio.h>
        int main(void)
 {
        printf("%e\n", 1234567.89);
        printf("%f\n\n\n", 1234567.89);
}
```

Output:

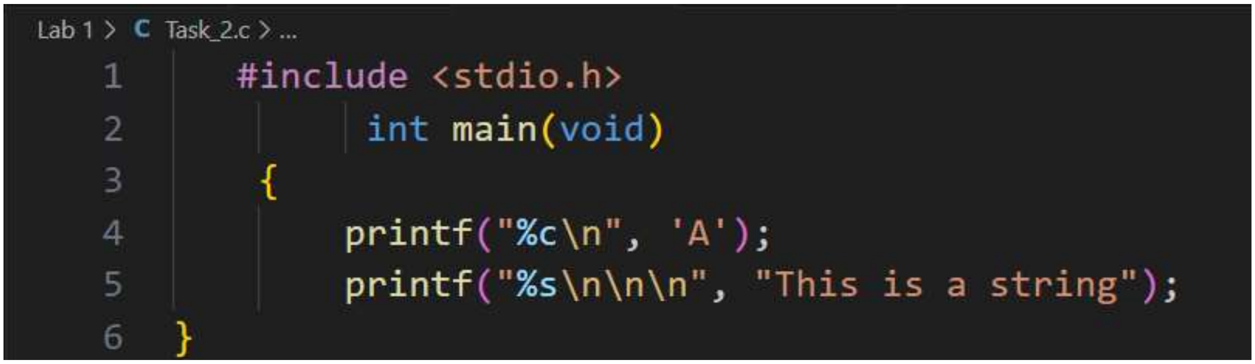


```
PS C:\Users\Infinix\Desktop\CSCI291 Labs\Lab 1> cd "c:\Users\Infinix\Desktop\CSCI291 Labs\Lab 1\" ; if ($?) { gcc Task_2.c -o Task_2 } ; if ($?) { .\Task_2 }
1.234568e+006
1234567.890000
```

<u>%c & %s</u>

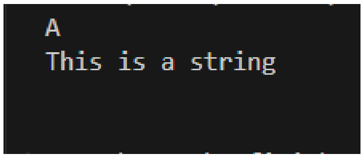
5- %c is a single character while %s is a string

Input:



```c
    #include <stdio.h>
        int main(void)
 {
        printf("%c\n", 'A');
        printf("%s\n\n\n", "This is a string");
}
```

Output:

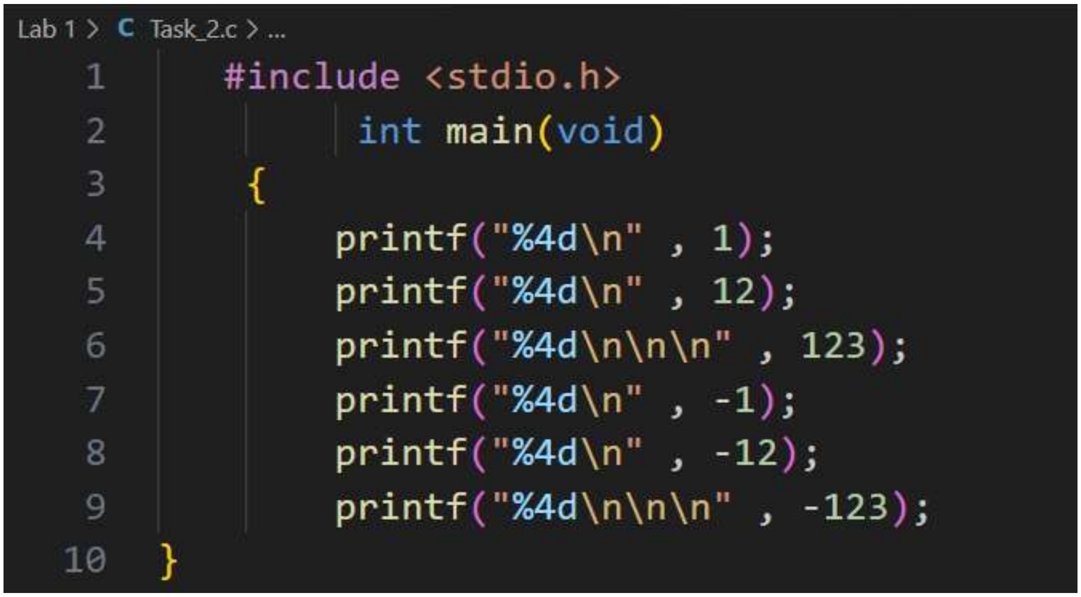


```
A
This is a string
```

%4d

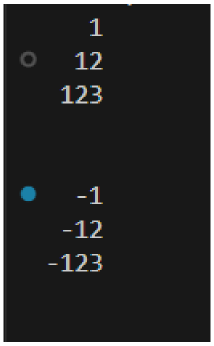
6- %4d prints an integer with a minimum width of 4 characters. Bear in mind that %4d will place a four digits space while neglecting how many digits the number is, as can be seen in the below screenshot.

Input:



```
   #include <stdio.h>
         int main(void)
    {
        printf("%4d\n" , 1);
        printf("%4d\n" , 12);
        printf("%4d\n\n\n" , 123);
        printf("%4d\n" , -1);
        printf("%4d\n" , -12);
        printf("%4d\n\n\n" , -123);
}
```

Output:

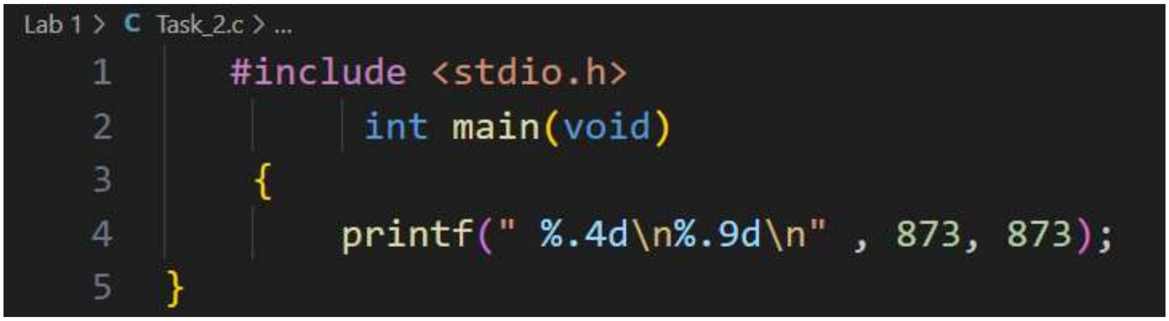


```
   1
  12
 123


  -1
 -12
-123
```

<u>%.4d & %.9d</u>

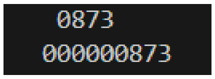
7- Prints the integer number padded with zeros to be minimum 4 digits long

Input:



```
    #include <stdio.h>
          int main(void)
      {
          printf(" %.4d\n%.9d\n" , 873, 873);
  }
```

Output:

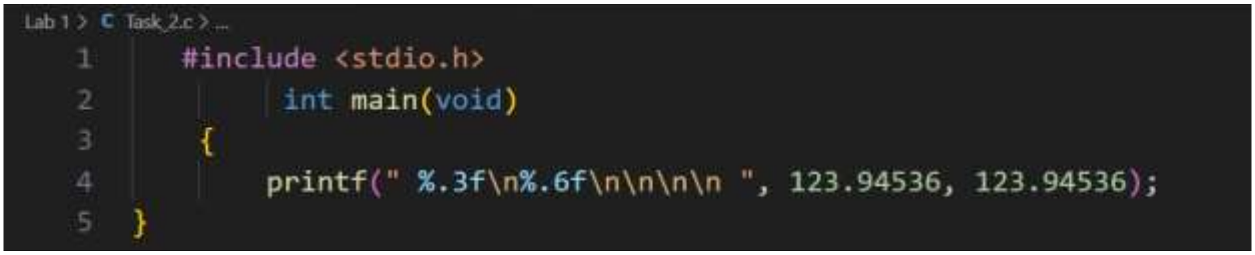


```
 0873
000000873
```

<u>%.3f & %.6f</u>

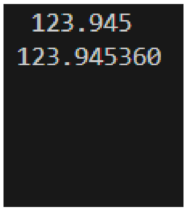
8- Prints floating-point numbers with specific decimal places. In this case, it prints a floating-point number with 3 decimal places and the other value has 6 decimal places.

Input:



```
    #include <stdio.h>
          int main(void)
      {
          printf(" %.3f\n%.6f\n\n\n\n ", 123.94536, 123.94536);
  }
```

Output:

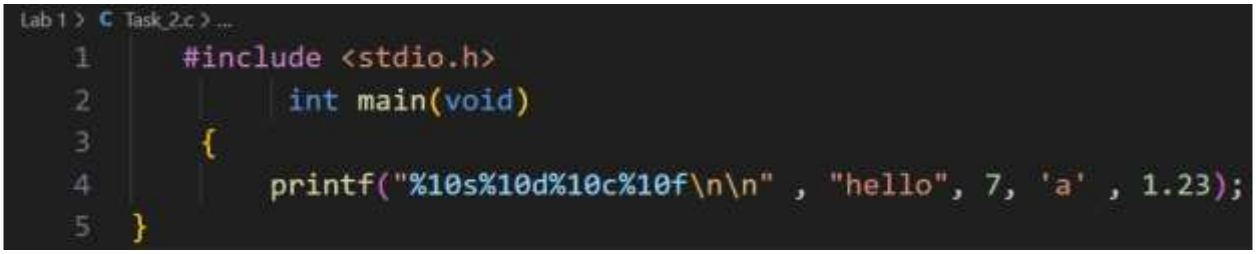


```
 123.945
123.945360
```

Alignment to the right of different data types

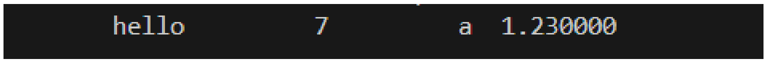
9- %10s is a string that is aligned 10 spaces to the right. %10d aligns 7 which is an integer 10 spaces to the right. %10c aligns a character 10 spaces to the right. %10f floating-point number 10 spaces to the right (remember float has 6 decimal places).

Input:



```
#include <stdio.h>
        int main(void)
    {
        printf("%10s%10d%10c%10f\n\n" , "hello", 7, 'a' , 1.23);
}
```

Output:

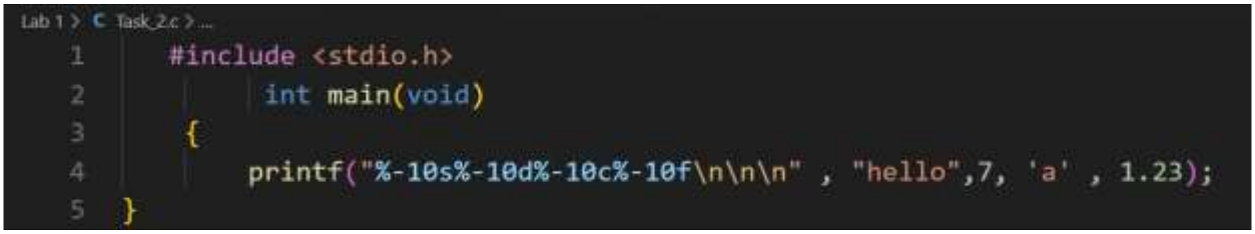


```
     hello         7         a  1.230000
```

Alignment to the left of different data types

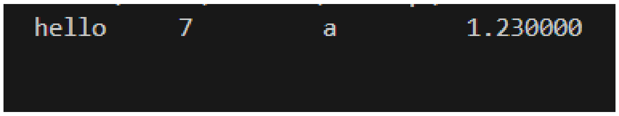
10- %-10s is a string that is aligned 10 spaces to the left. %-10d aligns 7 which is an integer 10 spaces to the left. %-10c aligns a character 10 spaces to the left. %-10f floating-point number 10 spaces to the left.

Input:



```
#include <stdio.h>
        int main(void)
    {
        printf("%-10s%-10d%-10c%-10f\n\n\n" , "hello",7, 'a' , 1.23);
}
```

Output:

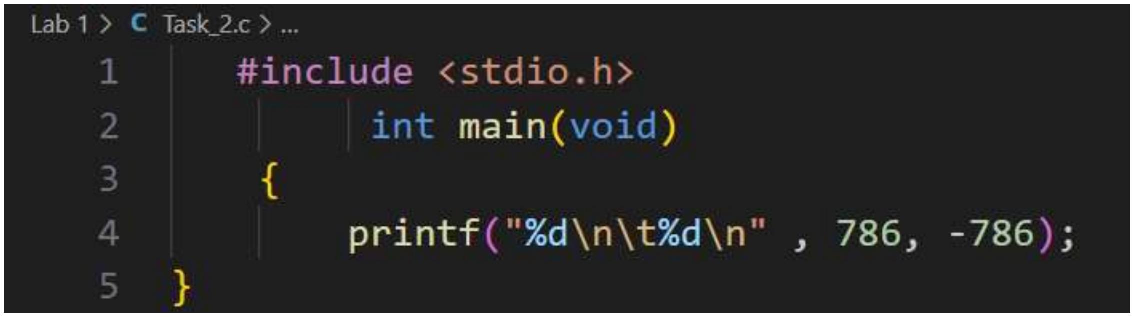


```
hello     7         a         1.230000
```

Print integers with/without a leading sign

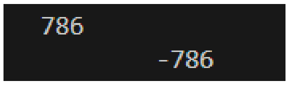
11- %d is the format specifier for integers. \t is a tab character, which adds a horizontal tab space.

Input:



```c
    #include <stdio.h>
            int main(void)
     {
        printf("%d\n\t%d\n" , 786, -786);
}
```

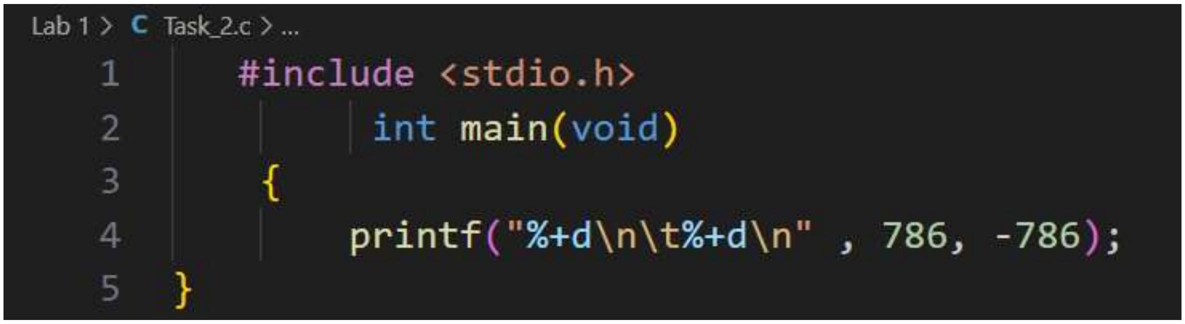
Output:



```
786
        -786
```

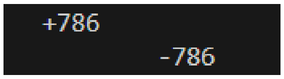
12- %+d makes sure that the sign of the integer is visible when the code is run.

Input:



```c
    #include <stdio.h>
            int main(void)
     {
        printf("%+d\n\t%+d\n" , 786, -786);
}
```

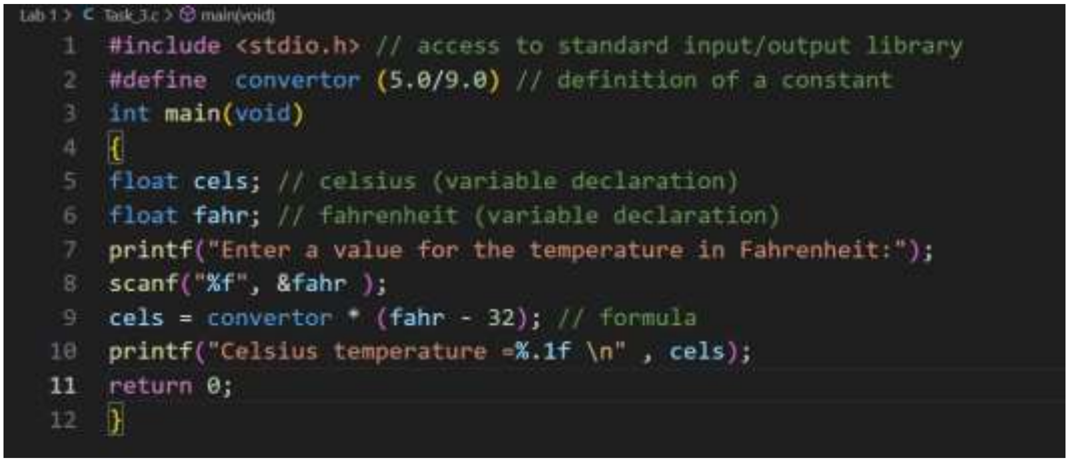
Output:



```
+786
        -786
```

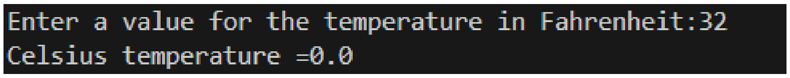
Task 3: Fahrenheit to Celsius program

Input:

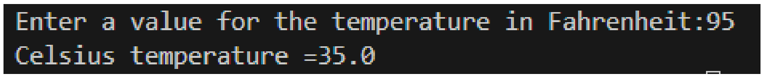


```c
#include <stdio.h> // access to standard input/output library
#define  convertor (5.0/9.0) // definition of a constant
int main(void)
{
float cels; // celsius (variable declaration)
float fahr; // fahrenheit (variable declaration)
printf("Enter a value for the temperature in Fahrenheit:");
scanf("%f", &fahr );
cels = convertor * (fahr - 32); // formula
printf("Celsius temperature =%.1f \n" , cels);
return 0;
}
```

Output:



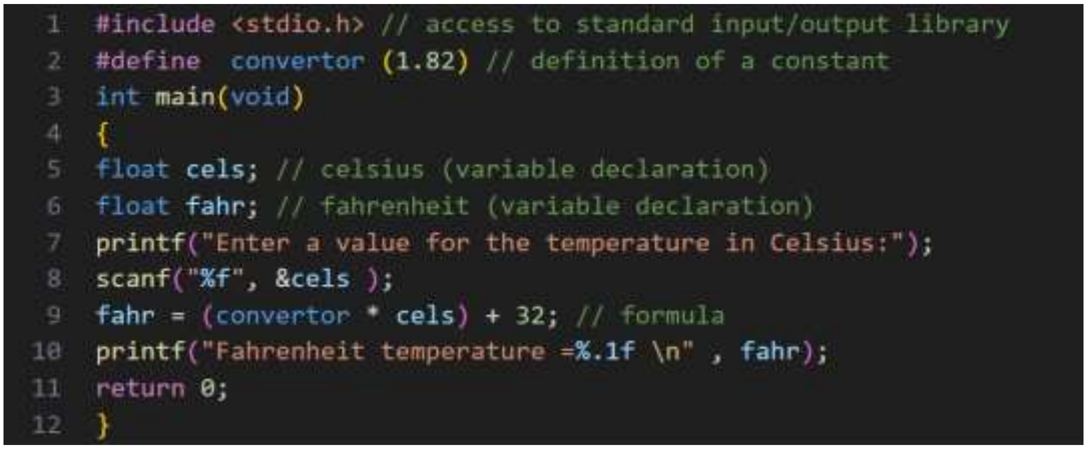
```
Enter a value for the temperature in Fahrenheit:32
Celsius temperature =0.0
```



```
Enter a value for the temperature in Fahrenheit:95
Celsius temperature =35.0
```

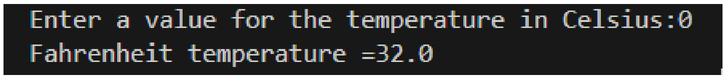
Task 4: Celsius to Fahrenheit program

Input:

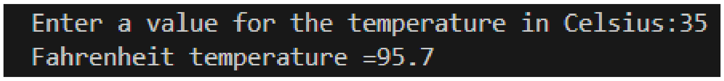


```c
#include <stdio.h> // access to standard input/output library
#define  convertor (1.82) // definition of a constant
int main(void)
{
float cels; // celsius (variable declaration)
float fahr; // fahrenheit (variable declaration)
printf("Enter a value for the temperature in Celsius:");
scanf("%f", &cels );
fahr = (convertor * cels) + 32; // formula
printf("Fahrenheit temperature =%.1f \n" , fahr);
return 0;
}
```

Output:



```
Enter a value for the temperature in Celsius:0
Fahrenheit temperature =32.0
```

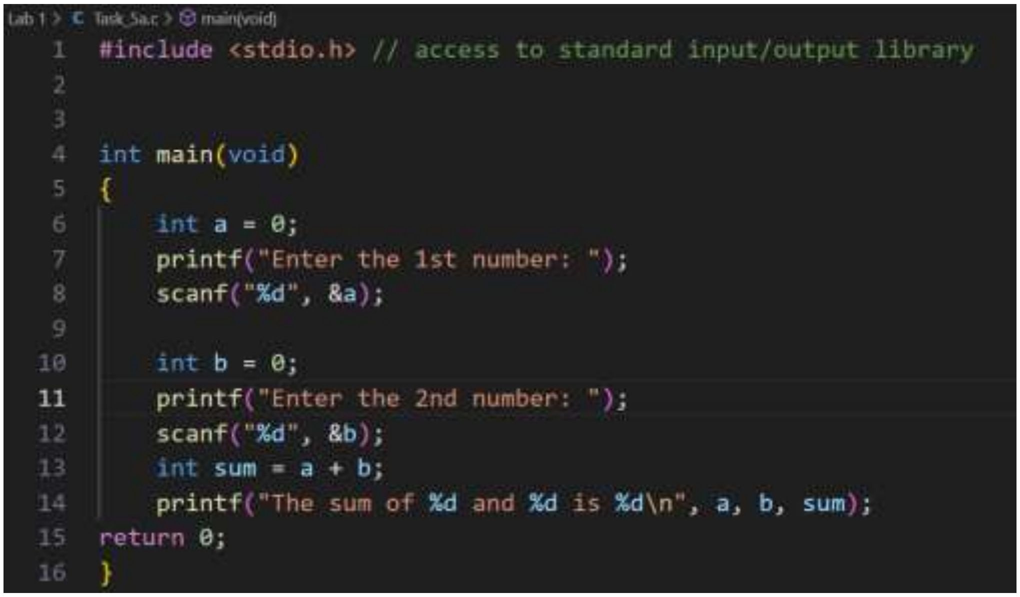


```
Enter a value for the temperature in Celsius:35
Fahrenheit temperature =95.7
```

Task 5: Basic I/O Operations and Mathematical Expressions

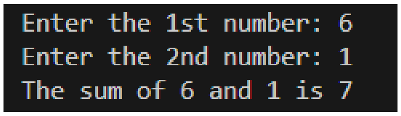
Part A: sum of two input integers

Input:



```c
#include <stdio.h> // access to standard input/output library


int main(void)
{
    int a = 0;
    printf("Enter the 1st number: ");
    scanf("%d", &a);

    int b = 0;
    printf("Enter the 2nd number: ");
    scanf("%d", &b);
    int sum = a + b;
    printf("The sum of %d and %d is %d\n", a, b, sum);
return 0;
}
```

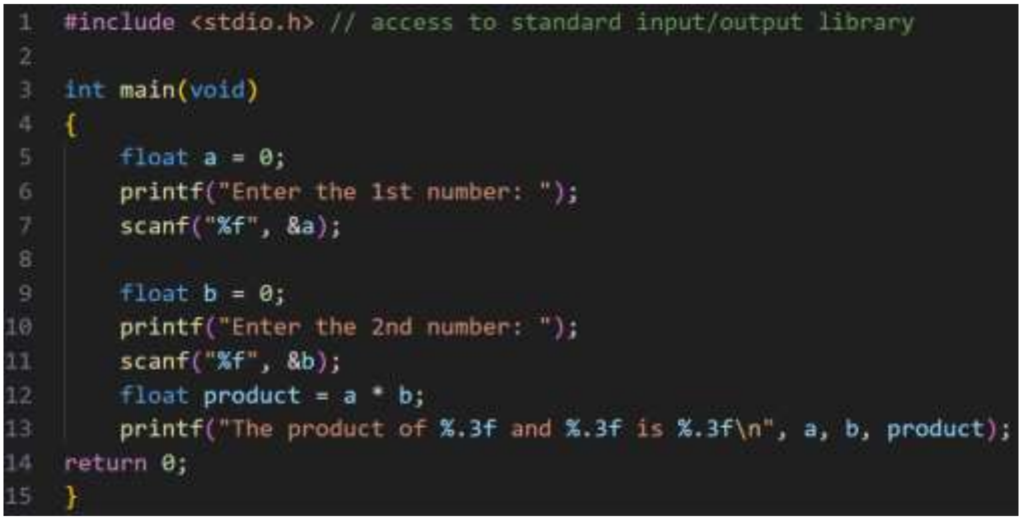
Output:



```
Enter the 1st number: 6
Enter the 2nd number: 1
The sum of 6 and 1 is 7
```

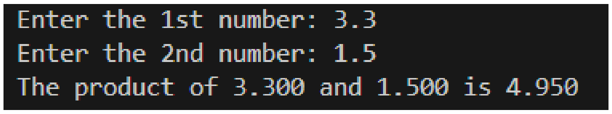
Part B: product of two input floats

Input:



```c
#include <stdio.h> // access to standard input/output library

int main(void)
{
    float a = 0;
    printf("Enter the 1st number: ");
    scanf("%f", &a);

    float b = 0;
    printf("Enter the 2nd number: ");
    scanf("%f", &b);
    float product = a * b;
    printf("The product of %.3f and %.3f is %.3f\n", a, b, product);
return 0;
}
```

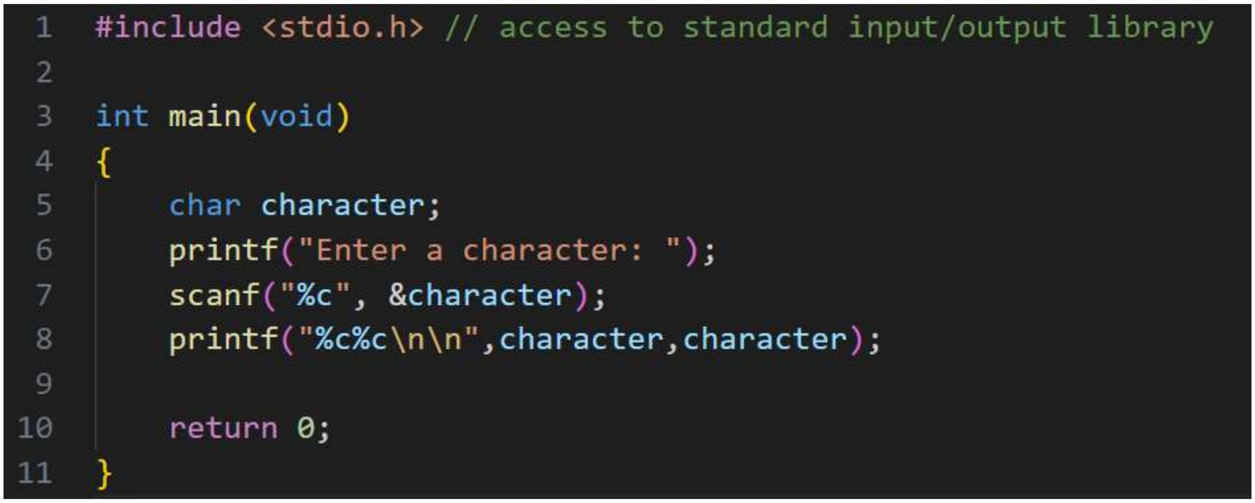
Output:



```
Enter the 1st number: 3.3
Enter the 2nd number: 1.5
The product of 3.300 and 1.500 is 4.950
```

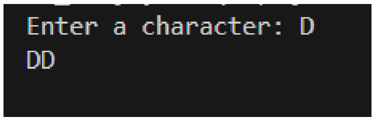
Part C: a character and prints it twice in the same row.

Input:



```c
#include <stdio.h> // access to standard input/output library

int main(void)
{
    char character;
    printf("Enter a character: ");
    scanf("%c", &character);
    printf("%c%c\n\n",character,character);

    return 0;
}
```

Output:

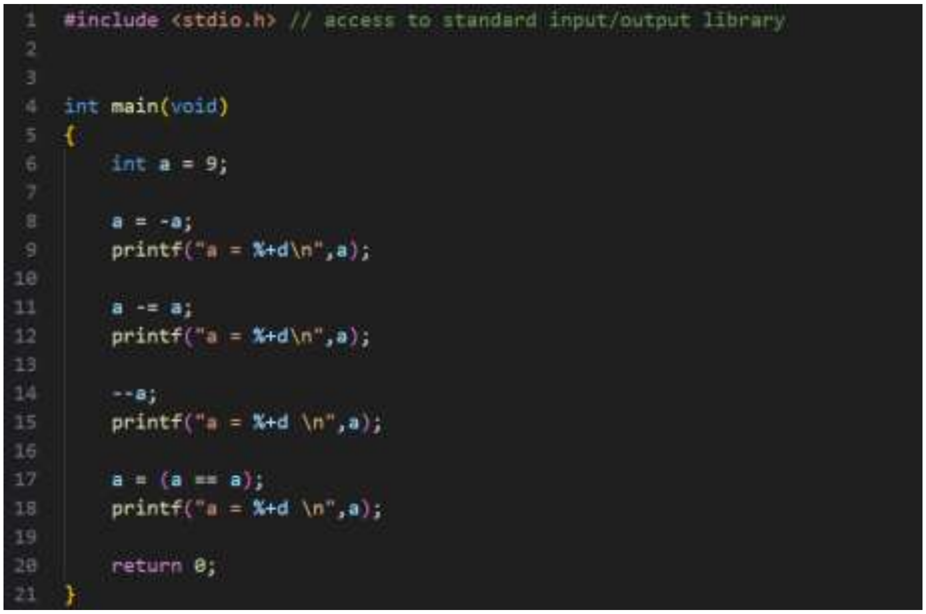


```
Enter a character: D
DD
```

<u>Part D: Write a C program with a variable integer initialized to 9 with the following statements:</u>

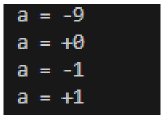
- a = - a;
- a - = a;
- --a;
- a = (a==a);

input:



```c
#include <stdio.h> // access to standard input/output library


int main(void)
{
    int a = 9;

    a = -a;
    printf("a = %+d\n",a);

    a -= a;
    printf("a = %+d\n",a);

    --a;
    printf("a = %+d \n",a);

    a = (a == a);
    printf("a = %+d \n",a);

    return 0;
}
```

Output:



```
a = -9
a = +0
a = -1
a = +1
```