Predict the output of following C program.

```c
#include<stdio.h>
int main()
{
    float x = 0.1;
    if (x == 0.1)
        printf("IF");
    else if (x == 0.1f)
        printf("ELSE IF");
    else
        printf("ELSE");
}
```

The output of above program is "***ELSE IF***" which means the expression "x == 0.1" returns false and expression "x == 0.1f" returns true.

Let consider the of following program to understand the reason behind the above output.

```c
#include<stdio.h>
int main()
{
   float x = 0.1;
   printf("%d %d %d", sizeof(x), sizeof(0.1), sizeof(0.1f));
   return 0;
}
```

The output of above program is "***4 8 4***" on a typical C compiler.

The values used in an expression are considered as double (double precision floating point format) unless a 'f' is specified at the end. So the expression "x==0.1" has a double on right side and float which are stored in a single precision floating point format on left side. In such situations float is promoted to double (see this). The double precision format uses more bits for precision than single precision format.
The binary equivalent of 0.1_{10} can be written as $(0.00011001100110011\ldots)_2$ which will goes up to infinity(See this article to know more about conversion). Since the precision of float is less than the double therefore after certain point(23 in float and 52 in double) it would truncate the result. Hence after promotion of float into double(at the time of comparison) compiler will pad the remaining bits with zeroes. Hence we get the different result in which decimal equivalent of both would be different. For instance,

```
In float
=> (0.1)10 = (0.0001100110011001100110011)2
In double after promotion of float ...(1)
=> (0.1)10 = (0.000110011001100110011000000000000000000...)2
                                        ^ padding zeroes here
In double without promotion ... (2)
=> (0.1)10 = (0.0001100110011001100110011001100110011001100110011
```

Hence we can see the result of both equations are different.
Therefore 'if' statement can never be executed.

```c
#include<stdio.h>
int main()
{
    float x = 0.5;
    if (x == 0.5)
        printf("IF");
    else if (x == 0.5f)
        printf("ELSE IF");
    else
        printf("ELSE");
}
```

Output:

```
IF
```

Here binary equivalent of 0.5_{10} is $(0.100000\ldots)_2$
(No precision will be lost in both float and double type). Therefore if compiler pad the extra zeroes at the time of promotion then we would get the same result in decimal equivalent of both left and right side in comparison(x == 0.5).