Programming Assignment on Int/FxP/FP numbers

Assume we have T-bit storage for a number. Write a C/C++ program for printing int, fixed-point (FxP) and IEEE-754 FP values for a particular combination or all combinations of T-bits. For example, if T=8, then there are a total of 2^8 (=256) combinations.

* For int, we have 1 sign bit and T-1 mantissa bits.

* For FxP, we have 1 sign bit, E integer bits and T-E-1 fraction bits.

* For FP, we have 1 sign bit, E exponent bits and T-E-1 mantissa bits.

Input formats: There are two input formats. Assuming the binary name is MyBinary, we will test:

Format 1: ./MyBinary T E Single ActualCombination

For example,

./MyBinary 8 3 Single 10101100

Here, output file should be named Rollnumber_T_E_Single_ActualCombination.txt

e.g., 1234567_8_3_Single_10101100.txt

Format 2: ./MyBinary T E All

For example,

./MyBinary 8 3 All

This should print all 256 combinations.

Here, output file should be named Rollnumber_T_E_All.txt

e.g., 1234567_8_3_All.txt

The integer is not stored in two's complement or one's complement form, but as a signed integer.

To ensure proper formatting, use this style for printing a single combination:

```
string s;
// read or generate s
std::cout << s <<setw(20) << getIntegerVal(s) << setw(20) << getFixedPoint(s) << setw(20) << getFP(s)<<"\n";
```

We will not test your program with any incorrect output or for T>32. We will also ensure (T-E-1)>=1.

Submission: The name of your submitted file MUST be Rollnumber_Formats.cpp or Rollnumber_Formats.c, e.g., 1234567_Formats.cpp 1234567_Formats.c (depending on whether you use C++ or C).

You need to upload your code as a single C++/C source file, which can be compiled and run with g++ without using any flags (except -lm for math library). It is OK to use c++11 features. Use of STL (e.g., vector, find, etc.) is acceptable.

Output: Your program should print everything in output file. Anything printed on terminal will be ignored. The TAs will test your code on different inputs. Please adhere to the above mentioned guidelines to reduce TA effort. Since we may use automated scripts to check your solution, not following the guidelines will lead to incorrect match and reduction in the marks.

If your code cannot be compiled, TAs will assess your program and give some marks. But 50% mark penalty will apply.

We are not concerned with the efficiency of your program but only its functionality, so do not worry if your code runs somewhat slow. We will also use plagiarism detection tool on all the submissions.

For checking your program yourself: see the attached binary, which shows the solution of the above program only when T<6 and E<4. You can run this to check for small values and to understand the output format we expect. For example, from its output you can see how NaN, normal, denormal and infinity are printed.

You can also check your program against

https://www.h-schmidt.net/FloatConverter/IEEE754.html by setting T=32 and E=8 and testing some values.