NAME: _____ GRADE: _____

EE442 / EE592 Real-Time Digital Signal Processing
Quiz #4
Allowed: Books, Code Printouts, Calculator, and Notes

Each question is worth 10 points unless otherwise noted.

Code listings of the files used in programming the TI DSP are listed at the end of this quiz.

1. In the TI pass code, we call process_signal() from within dsk_app.c every sample period. Please explain why the call has to be

```
process_signal(short inputRight, short inputLeft, short *outputRight, short *outputLeft)
```

instead of

```
process_signal(short inputRight, short inputLeft, short outputRight, short outputLeft)
```

2. In the following code, please explain why the 15 right shifts are required for proper computation of z when we assume a fixed-point fractional form.

```
short x, y;
int z;

void main (void)
{
      x = 16384; // 0.5 in fixed point fractional form
      y = 16384;
      z = (x * y)>>15;
}
```

3. Consider a FIR filter with the following coefficients

$$h[n] = \begin{cases} 0.5, & n = 0 \\ 0.25, & n = 1000 \\ 0.125, & n = 2000 \\ 0, & \text{otherwise} \end{cases}.$$

Explain why it would be more efficient to implement the filter using tap() instead of cfir().

4. In all of the DSP routines involving a circular array to store filter states, the array w and the oldest filter state p are passed as input arguments. For example

```
short cfir(short M, short *h, short *w, short **p, short x)
```

is called in process_signal() with

```
*outputRight = cfir(FILTERORDER, coeffs, states, &oldestStatePtr, inputRight);
```

Please explain why

```
short cfir(short M, short *h, short *w, short *p, short x)
```

and

```
*outputRight = cfir(FILTERORDER, coeffs, states, oldestStatePtr, inputRight);
```

will not properly implement a circular array.

5. Please explain the wrap() function:

```
void wrap(short M, short *w, short **p)
{
    if (*p < w || *p > (w + M))
        *p = w + (*p - w) % (M + 1);
    if (*p - w < 0)
        *p += M+1;
}
```

where M is the array order, *w is a pointer to the base address of the circular array, and **p is a pointer to a pointer to the array (called by reference).

Write a C code for the following *real-time* programming tasks (Questions 6 – 7), assuming the TI dsk_app passcode. Identify by file name (user_data.h, initialize_program.c, process_signal.c) and line number where you would place your instructions. Printouts of the files are listed at the end of this quiz. The first programming task is done as an example.

6. Write a code which multiplies the right channel by $g = 0.7$.

Solution:

In user_data.h insert at line 12 the following line:

```
#define G 22938 /* short equiv for 0.7 */
```

In process_signal.c replace line 12 with the following line:

```
*outputRight = cround(G * inputRight)>>15;
```

7. Implement (real-time) the FIR filter in Problem 3 for the right channel using tap().

Solution:

EE592 Only

EE442 students: if time permits, you may wish to try these problems, however, no additional points will be earned.

8. Write a C code (non-real-time) that bit reverses the first N bits in x and stores the stores the result in y. Note that $1 \le N \le 16$.

```
void main (void)
{
    short N = <INITIALIZED BY USER>;
    short x = <INITIALIZED BY USER>;
    short y;

<YOUR CODE HERE>

    return 0;
}
```

Code Listings

```
1  /*
2  * USER_DATA.H
3  */
4
5  #ifndef USER_DATA
6  #define USER_DATA
7
8  void initialize_program();
9  void process_signal(short inputRight, short inputLeft, short *outputRight, short
       *outputLeft);
10
11 /* Global variables here as extern, initializations in initialize_program.c */
12
13 #include "DSPFunctionsFixedPoint/util.h"
14
15 #endif
```

```
1  /*
2  * INITIALIZE_PROGRAM.C
3  */
4
5  #include "user_data.h"
6
7  /***************************************/
8  /* Global variable initializations here */
9  /***************************************/
10
11 void initialize_program()
12 {
13
14 }
```

```
1  /*
2  * PROCESS_SIGNAL.C
3  */
4
5  #include "user_data.h"
6
7  void process_signal(short inputRight, short inputLeft, short *outputRight, short
       *outputLeft)
8  {
9  /*******************************/
10 /* Process right channel sample */
11 /*******************************/
12 *outputRight = inputRight;
13
14 /*******************************/
15 /* Process left channel sample */
16 /*******************************/
17 *outputLeft = inputLeft;
18 }
```