Name: _____ **RCS ID:** _____ @rpi.edu

♡ ♡ ♡ CSCI 2500 — Computer Organization ♡ ♡ ♡
Fall 2018 Quiz 2 (September 28, 2018)

Please silence and put away all laptops, notes, books, phones, electronic devices, etc. This quiz is designed to take 25 minutes; therefore, for 50% extra time, the expected time is 38 minutes and 100% extra time is 50 minutes. Questions will not be answered except when there is a glaring mistake or ambiguity in a question. Please do your best to interpret and answer each question.

1. **(5 POINTS)** Which of the following is **not** one of the four MIPS design principles? Clearly circle the **best** answer.

   (a) Simplicity favors regularity
   (b) Smaller is faster
   (c) Make the common case fast
   (d) Good design demands good compromises
   (e) `vi` is orders of magnitude better than `emacs`

2. **(5 POINTS)** In C, for function `snapchat()`, we want to pass first parameter `x` (an `int`) by value and second parameter `y` (an `int`) by reference. To do so, we should use the following function prototype. Clearly circle the **best** answer.

   (a) `int snapchat( int x, int * y );`
   (b) `int snapchat( int x, int & y );`
   (c) `int snapchat( int & x, int y );`
   (d) `int snapchat( int * x, int y );`
   (e) `int snapchat( int & x, int & y );`
   (f) `int snapchat( int * x, int * y );`

3. **(5 POINTS)** To use MIPS to access data out in memory (i.e., not in a register), you use the *Load Word* (`lw`) instruction. Assuming array `A` has its base address in register `$s5`, translate the C code snippet below into valid MIPS code. Clearly circle the **best** answer.

```
int A[100];    /* array of 100 integers */
int s1 = A[4];
```

   (a) `lw $s5,4($s1)`
   (b) `lw $s5,16($s1)`
   (c) `lw $s1,4($s5)`
   (d) `lw $s1,16($s5)`
   (e) `lw $s1,$s5,4`
   (f) `lw $s1,$s5,16`

4. **(15 POINTS)** Translate the given C code into MIPS assembly code. You are only allowed to hard-code the initial value of variable `y`. All other values must be calculated using `add` and `sub` instructions. Do **not** simplify the expressions shown in the given C code. Use only valid registers `$s0` to `$s7` and `$t0` to `$t9`. Place your final result for variable `x` into register `$s0`.

```
int x, y = 128, z;
z = y + y;
x = ( y - z ) - ( z + y )
```

5. **(30 POINTS)** The C code below reads one line of input, then produces one line of output. Given a 64-bit architecture and an input of "`007QRSTUVWXYZ567`" (terminated with a newline '`\n`' character), what is the **exact** terminal output? Note that this code compiles and executes without crashing; further, `#include` directives are not shown.

```
int main()
{
  int m = 2;
  char line[128];
  fgets( line, 128, stdin );
  char * t = line + 3;
  printf( "%lu-[", sizeof( char * ) );
  while ( *t )
  {
    if ( isalnum( *t ) ) printf( "%*c", m, *t );
    t += m;
  }
  printf( "]-%lu%d\n", sizeof( int ), m );
  return EXIT_SUCCESS;
}
```

6. **(40 POINTS)** As with Quiz 1, the `vowels()` function below is supposed to take input string `s` and return a new string containing only the vowels in `s`. All other characters should be ignored. As an example, if `s` is "`SnApChAt RuLeS!!!`" then the function returns "`AAue`" (i.e., a string of four bytes). Unfortunately, there are four errors in the code below. Find and correct each error. And do not simply rewrite the function or change the coding style used.

```
char * vowels( char * s )
{
  int count = 0;
  char * i, * j, * k, * v = "aeiouAEIOU";

  for ( i = s ; *i ; i++ )
    for ( j = v ; *j ; j++ )
      if ( *j == *i )
        count++;

  char * result = calloc( count, sizeof( char ) );
  for ( i = s, k = result ; *i ; i++ )
    for ( j = v ; *j ; j++ )
      if ( *j = *i )
        *k = *i;

  return *result;
}
```