Name: ____________________________ **RCS ID:** ___________ @rpi.edu

♢ CSCI 2500 — Computer Organization ♢
Fall 2018 Quiz 1 (September 14, 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 the statement of a question. Please do your best to interpret and answer each question.

1. **(5 POINTS)** When you use `malloc()` or `calloc()` to dynamically allocate memory, where is the memory allocated? Clearly circle the **best** answer.

   (a) The runtime stack

   [✓ (b) The runtime heap]

   (c) The runtime code segment

   (d) The runtime error

   (e) The runtime text segment

   (f) The runtime environment

2. **(5 POINTS)** In C, to pass variable `x` by value and variable `y` by reference to function `snapchat()`, use the following syntax. Clearly circle the **best** answer.

   (a) `snapchat( x, *y );`

   (b) `snapchat( *x, y );`

   [✓ (c) `snapchat( x, &y );`]

   (d) `snapchat( &x, y );`

   (e) `snapchat( x, y );`

   (f) `snapchat( *x, *y );`

3. **(5 POINTS)** In the shell (i.e., a terminal), to determine the process exit code for the last process that was executed, use the following command. Clearly circle the **best** answer.

   (a) `echo $$`

   (b) `echo $-`

   [✓ (c) `echo $?`]

   (d) `echo $EXIT_CODE`

   (e) `echo $!`

   (f) `echo $SNAPCHAT`

4. **(15 POINTS)** Given `unsigned int` variable `n`, write exactly one `malloc()` call to dynamically allocate the exact amount of memory needed to create an array containing exactly `n` `double` values. Call this array `a` and be sure to properly declare this variable.

```
double * a = malloc(n*sizeof(double));
```

5. **(30 POINTS)** What is the **exact** terminal output of the C code below? Note that this code compiles and executes without crashing; further, `#include` directives are not shown.

```
int main()
{
  int x = 10;
  int * p = &x;
  char * s = calloc( x, sizeof( char ) );
  char * t;
  printf( "AA%d%lu-%lu%dBB\n", x, sizeof( short ), sizeof( int * ), *p );
  strcpy( s, "LLMMNNOO" );
  t = s + 5;
  *t = 'Z';                         AA102-810BB\n (no need to write '\n')
  printf( "%s", t );                ZOOLLM
  t--;
  t--;
  *t = '\0';
  printf( "%s\n", s );
  free( s );
  return EXIT_SUCCESS;
}
```

6. **(40 POINTS)** 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 i, j, k, count;   count = 0
  char * result;
  char * v = "aeiouAEIOU";

  for ( i = 0 ; i < strlen( s ) ; i++ )
    for ( j = 0 ; j < strlen( v ) ; j++ )
      if ( s[i] == v[i] )  s[i] == v[j]
        count++;

  result = malloc( count );  malloc(count + 1)
  for ( i = 0, k = 0 ; i < strlen( s ) ; i++ )
    for ( j = 0 ; j < strlen( v ) ; j++ )
      if ( s[i] == v[j] )
        result[k++] = s[i];
  result[k] = "\0";   result[k] = '\0';
  return result;
}
```