ECE 2220: System Programming Concepts **Fall 2016**
Problem Set 3 **Due: in class, Wednesday, September 28**

Assigned reading: Hoover, Chapter 3. For Chapter 4, this homework set covers sections 4.1 and 4.2. We will cover the remainder of chapter 4 on the next homework set (and the first exam will test material through section 4.2 only). Each problem is worth 10 points

From Chapter 3, starting on page 94

1. Number 13

From Chapter 4, starting on page 127

2. Number 1
3. Number 2
4. Number 3
5. Number 4
6. Correct the following code errors by drawing a line through the erroneous statement and writing a corrected statement on the same line. All versions should print "x^2=169". The only change you can make to the code is the placement of "`*`" and "`&`" symbols. No other changes are permitted.

```
void square(int *p) {
      *p = *p * *p;
}
```

Part (a)

```
      void main(void) {
            int x, *p;
            p =  x;
            x =  13;
            square( p )
            printf("x^2=%d\n",  x);
      }
```

Part (b)

```
      void main(void) {
            int x, *p;
            x =  13;
            p =  &x;
            square( &p )
            printf("x^2=%d\n",  x);
      }
```

Part (c)

```
      void main(void) {
            int x;
            x =  13;
            square( *x )
            printf("x^2=%d\n",  x);
      }
```

7. Give the memory map for the following code. If the value at a memory location is changed, cross out the old value and append the new value in the figure such that all values assigned to a memory location can be seen.

```
int array[5], what[5];
int *ptr;
int offset, j;

for (j=0; j<5; j++)
      array[j] = 2 * j + 10;

ptr = &(array[0]);
offset = 4;

what[0] = *(ptr + offset);
what[1] = ptr[offset - 3];
what[2] = *(array + offset);
what[3] = *ptr+offset;
what[4] = *ptr++;
```

8. The following code needs a function called **`stats()`**. The function takes a pointer to an array of integers and returns the average value of the numbers in the array. In addition, a side effect of the function is that the values **`array_max`** and **`array_min`** will contain the largest and smallest value in the array after the **`stats`** function is called. You must write the **`stats()`** function, including the arguments that are passed to the function. Other than how the **`stats()`** function is called no other changes to **`main()`** are permitted. No global variables are permitted.

```
//  assume this file is called my_prog.c, and you make an executable file
//  called my_prog.  It is run using    ./my_prog 1000

   #include <stdlib.h>
   #include <stdio.h>
   int main(int argc, char *argv[])
   {
      int *array;
      int array_max, array_min, array_size;
      double array_average;
      int j;

      array_size = atoi(argv[1]);
      array = (int *) malloc( array_size * sizeof(int));

      // fill the array with random numbers
      for (j=0; j < array_size; j++)
            array[j] = array_size * drand48();

      array_average = stats(       // you must fill in details here

      printf("Avg=%g, Max=%d, Min=%d\n",
             array_average, array_max, array_min);

      free(array);
      return 0;
   }
```