

COMP 166 – Week 10 Quiz

Name: _____

1. Fill in the blanks in the following code fragment to make it count all odd integers in an array. Assume `p` is declared as an `int` pointer and that it already points to the first value in some array. You don't know how many values are in the array, but the loop stops when it encounters the value zero (0) the first time. You may not use any variables other than `p` and `count`, or add any new lines of code.

```
int count = 0;

while ( _____ != 0) {

    if ( _____ % 2 == 1) {

        count = count + 1;

    }

    _____ = _____ + 1;

}

printf("Number of odd values: %i\n", count);
```

2. Complete the following function. It checks the value that each parameter points to (i.e. is dereferenced to), and returns the *address* that the smaller value is stored at. For example, suppose `a` is stored at address 1000 and has a value of 10, and `b` is stored at address 1004 and has a value of 5; then this function returns the address of `a`, which is 1000.

```
int * smaller(int * a, int * b) {

}

}
```

COMP 166 – Week 10 Quiz

3. Write a small code fragment that *dynamically* allocates space for one `double` value and assigns 16.5 into the new space. (Assume the memory allocation succeeds; you don't have to handle the case where there is not room for the `double` value in RAM.)

4. Suppose a variable “p” has just been declared and assigned a value returned from `malloc()`. Write a small code fragment that checks to see if the memory allocation succeeded, and prints “Out of memory.” to the screen if it did not. (Your code does not have to do anything if the allocation succeeded.)

Assume your code will be placed immediately after the call to `malloc()`.

5. Consider the following 2-D array declaration:

```
int m[6][4];
```

What is the 1-dimensional *offset* of the element `m[4][2]`? Offset: _____

6. Which of the following correctly allocates space for a 2-D array of integers *dynamically* that contains 15 rows and 32 columns, and assigns the result to an appropriate variable?

- (a) `int m[][] = (int) malloc(15,32);` (c) `int * m = (int*) malloc(15*32*sizeof(int));`
(b) `int m[15][32];` (d) `int * m = (int*) malloc(15*32, sizeof(int*));`