

COMP 166 – Week 11 Quiz

Name: _____

1. Write a small code fragment that dynamically allocates space for one `double` value and assigns 7.0 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.)

2. Which of the following code fragments *define* a new struct called “Point” that contains two double values (`x` and `y`)? The correct code would appear in the corresponding header file; you are not *using* the structure, just defining it.

(a)

```
#define struct Point {  
    double x, y;  
}
```

(c)

```
struct Point p1 = {double x = 7, double y = 13};
```

(b)

```
struct Point {  
    double x, y;  
};
```

(d)

```
Point.x = 7;  
Point.y = 13;
```

3. Assume the `Point` structure has been properly defined as in **Question 2**.

TRUE or FALSE: The following code fragment creates two `Point` objects: one using *static* allocation, and assigning to a *regular variable* called “p1”; and one using *dynamic* allocation, and assigning to a *pointer* called “p2”.

```
struct Point p1;  
struct Point * p2 = &p1;
```

TRUE

FALSE

COMP 166 – Week 11 Quiz

4. Suppose that you have two variables, `p1` and `p2`. Both of these variables refer to different `Point` objects as defined in **Question 2**; however, `p1` is a regular reference (*i.e.* not a pointer), and `p2` is a pointer to the object. Which of the following code fragments correctly assigns the values `3.7` and `12.1` to the “`x`” members of `p1` and `p2`, respectively?

- (a) `struct p1.x = 3.7;`
`struct p2.x = 12.1;`
- (b) `p1.x = 3.7;`
`p2.*x = 12.1;`
- (c) `*p1.x = 3.7;`
`**p2.x = 12.1;`
- (d) `p1.x = 3.7;`
`p2->x = 12.1;`

5. Which of the following correctly allocates space for a 2-D array of `chars` *dynamically* that contains 12 rows and 8 columns, and assigns the result to an appropriate variable?

- (a) `char * m = (char*) malloc(12, 8, sizeof(char*));`
- (b) `char * m = (char*) malloc(12*8*sizeof(char));`
- (c) `char m[12][8];`
- (d) `char m[][] = (char) malloc(12,8);`

6. TRUE or FALSE: When a `struct` is passed to a function as a regular parameter (*i.e.* not as an explicit pointer), the entire content of the structure is copied to the function, and changes within the function do *not* affect the original object.

TRUE

FALSE