1. What is the difference between a struct and a class? Why might you want to use one over the other? How do you access the data members for each one? Give an example of a simple struct. Also, provide a brief example of the correct method for accessing a data member for both options.

```
struct -> default is public (data members)
   -> Employee, name
class  -> default is private (data)
   -> Employee, get_name() / set_name()
```

2. In the following snippets, the focus is on the use of the * and & operators. Next to each code snippet, provide a brief explanation of what the operator(s) mean for the given scenario.

a. `int *p;` declaring a pointer.
b. `p = &num;` assigns address of num to p.
c. `void function (double &num){` pass by reference
   `num = 4; }
```
d. `void function (double *num){` pass-by-pointer
   `num = 4; }` dereference

e. `void function (double &num){` memory leak
   `num = 4; }
```
f. `void function (double &num){` array
   `num = 4; }
```
g. `void function (double *array){` read-only
   `// code here}
```
h. `void function (double **array){` 2 array / pointer passed-by-pointer
   `// code here}
```
i. `void function (const double *array){` ERROR!
   `array[0] = 3.5; }`
```

3. Dynamic Arrays! First, what is a dynamic array? Write the line of code to create a 1d dynamic array that allocates memory for 40 data elements of type char.

```
char *array = new char[40];
```

4. Now, write code (logically, ok if syntax is not 100%) for a function named `create_array()` that allocates memory for a dynamic 2d array. The function should not have any return value, and should have parameters for each dimensions size (rows and columns) as well as a parameter for the pointer to your array. What is the data type of the array parameter?

```
void create_array (int i, int j, int **&p) {
    p = new int*[i]; // 1) array of int*
    for (int r=0; r<i; r++)
        p[r] = new int[j]; // 2) array of int
```

5. Extra: As a group, write the `delete_array()` function to free all allocated memory from code demo example!

```
void delete_array (int **&p, int row, int col) {
    for (int i=0; i<row; i++)
        delete [] p[i]; // delete arrays of int
    delete [] p; // delete array of int *
```
int **cp

[Diagram of arrays and matrices]

p[1][2]