1. (6 pts) What is a one-dimensional array in C? Show an example, including a memory snapshot to illustrate your answer.

2. (4 pts) Explain the distinction between an element of an array and its corresponding array subscript.

3. Write C statements to accomplish each of the following tasks. Assume these statements will be executed in the order shown.
   3.a. (3 pts) Declare an array named `x` that can hold 100 `double` values. Do not initialize the values.
3.b. (3 pts) Set the value of the first element of the array $x$ to $\text{NUM}$ (assuming $\text{NUM}$ is a symbolic constant defined elsewhere in your code).

3.c. (6 pts) Write a for loop that sets each remaining value of $x$ to the value of the preceding element plus $0.1$ (i.e. second element is first + $0.1$, third is second + $0.1$, etc.).

3.d. (3 pts) Print the value of the last element of $x$.

4. (9 pts) In C, arrays can be initialized as with the following example, where $\text{NUM}$ is an int constant:

```c
int arr[ NUM ] = { integer_0, integer_1, ..., integer_N };
```

If $\text{NUM}$ is omitted, the length of the array is set to the number of values specified. In the above example, the length of $\text{arr}$ would be $N + 1$ if $\text{NUM}$ were omitted. If $\text{NUM}$ is less than the number of elements specified, the compiler will issue a warning. If $\text{NUM}$ is greater than the number of elements specified, then all remaining elements (i.e. elements at indices greater than $N$) will be initialized to $0$. Note that this will be $0$ at the binary representation, regardless of the data type of the array (i.e. an array of char will be initialized to ‘\0’, an array of pointers will be initialized to NULL, and so on).

It is also possible to simply declare arrays without initialization, like so:

```c
int arr[ NUM ];
```
For arrays that are declared without initialization, the value of all elements is undefined.

Consider the following code:
```c
int arr1[10] = { 4, 5, 6 };
int arr2[10];
```

What is the value of each of the following expressions? Note that the correct answer may be undefined. 1 pt apiece.

a. `arr1[0]`

b. `arr1[1]`

c. `arr1[2]`

d. `arr1[3]`

e. `arr1[9]`

f. `arr1[10]`

g. `arr1[11]`

h. `arr1[-1]`

i. `arr2[5]`

5. (10 pts) Write a `void` function named `printReverse` that takes one `double` array named `d` and one `int` named `n` as
arguments, where \( n \) is the number of elements in \( d \). This function must print the \( n \) array values in reverse order, i.e. the last element of \( d \) should be printed first and the first element of \( d \) should be printed last. Separate the printed values by newlines.

Pre-lab End. Adapted from Michael Costanzo by Kyle Dewey.