SCHOOL OF INFORMATICS, COMPUTING, AND ENGINEERING

INDIANA UNIVERSITY

WILL BOLAND

11/15

1. Answer each of the following:
   a) A pointer variable contains as its value the **memory address** of another variable.
   b) The three values that can be used to initialize a pointer are **NULL**, **0,** and **Memory address (hex).**
   c) The only integer that can be assigned to a pointer is **0**.

2. State whether the following are *true* or *false*. If the answer is *false*, explain why.

   - a) The address operator (&) can be applied only to constants, to expressions and to variables declared with the storage-class register.
     - **False: constants, expressions, variables of all types, pointers**
   - b) A pointer that's declared to be void can be dereferenced.
     - **False: must be type cast**

- c) Pointers of different types may not be assigned to one another without a cast operation.
  - **True**

3. Answer each of the following. Assume that single-precision floating-point numbers are stored in 8 bytes, and that the starting address of the array is at location 2030100 in memory. Each part of the exercise should use the results of previous parts where appropriate.

- a) Define an array of type float called numbers with 5 elements, and initialize the elements to the values 0.11, 0.22, 0.33, ..., 0.55. Assume the symbolic constant SIZE has been defined as 5.
  - **float numbers[5] = {0.11, 0.22, 0.33, 0.44, 0.55};**
- b) Define a pointer, nPtr, that points to an object of type float
  - **float* nPtr = numbers;**
- c) Print the elements of array numbers using array subscript notation. Use a for statement and assume the integer control variable i has been defined. Print each number with 2 position of precision to the right of the decimal point.

```
for(i = 0; i < numbers.SIZE; i++) {
    printf("%.2f\n", numbers[i]);
}
```

- d) Give two separate statements that assign the address of last element of array numbers to the pointer variable nPtr.
  - **&numbers[4] = nPtr;**
  - **&numbers[4] = *&nPtr;**
- e) Print the elements of array numbers using pointer/offset notation with the pointer nPtr.

```
int i;
for(i = 0; i < numbers.SIZE; i++) {
   printf("%.2f\n", *(nPtr + i));
}
```

- f) Print the elements of array numbers using pointer/offset notation with the array name as the pointer.

```
int i;
for(i = 0; i < numbers.SIZE; i++) {
   printf("%.2f\n", *(numbers + i));
}
```

- g) Print the elements of array numbers by subscripting pointer nPtr.

```
int i;
for(i = 0; i < numbers.SIZE; i++) {
   printf("%.2f\n", nPtr[i]);
}
```

- h) Refer to element 2 of array numbers using array subscript notation, pointer/offset notation with the array name as the pointer, pointer subscript notation with nPtr and pointer/offset notation with nPtr.
  - **numbers[1]; //array subscript notation**
  - ***(numbers + 1); //offset notation with array name**
  - **nPtr[1]; //pointer subscript notation**
  - ***(nPtr + 1); //pointer offset notation**
- i) Assuming that nPtr points to the end of array numbers (i.e., the memory location after the last element of the array), what address is referenced by nPtr - 5? What value is stored at that location?
  - **2030100**
  - **0.11**
- j) Assuming that nPtr points to numbers[5], what address is referenced by nPtr -= 2? What's the value stored at that location?
  - **2030124 //decimal representation of address**
  - **0.44**

4. For each of the following, write a statement that performs the indicated task. Assume that floating-point variables number1 and number2 are defined and that number2 is initialized to 5.3.

- a) Define the variable fPtr to be a pointer to an object of type float.
  - **float* fPtr;**

- b) Define the variable fPtr2 to be a pointer to a pointer to an object of type float.
    - **float* fPtr2;**
- c) Assign the address of variable number2 to pointer variable fPtr.
    - **fPtr = &number2;**
- d) Print the value of the object pointed to by fPtr.
    - **printf(“%f”, *fPtr);**
- e) Assign the value of the object pointed to by fPtr to variable number1.
    - **number1 = *fPtr;**
- f) Assign the address of variable fPtr to pointer variable fPtr2.
    - **fPtr2 = &fPtr1;**
- g) Print the value of number1.
    - **printf(“%f”, number1);**
- h) Print the address of number2. Use the %p conversion specifier.
    - **printf(“%p”, &number2);**

- i) Print the address stored in fPtr. Use the %p conversion specifier. Is the value printed the same as the address of number2?
    - **printf("%p", fPtr);**
    - **Yes.**
- j) Print the address stored in fPtr2. Use the %p conversion specifier.
    - **printf("%p", fPtr2);**

5. Do each of the following:
    - a) Write the **function header** for a function called myswap that takes two pointers to integer numbers x and y as parameters and returns a Boolean value.
        - int myswap(int* x, int* y) //header (not definition)
    - b) Write the function prototype for the function in part (a).

int myswap(int* x, int* y); //function prototype