- CPSC 275: Introduction to Computer Systems

CPSC 275: Introduction to Computer Systems

Fall 2025

- Syllabus
- Schedule
- Resources
- Upload
- Solution

Homework 9

NOTE: You are not required to hand in the following exercises, but you are strongly encouraged to complete them to strengthen your understanding of the concepts covered in class.

1. 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 `number1` is initialized to 7.3.
   A. Define the variable `fPtr` to be a pointer to a `float`.
   B. Assign the address of variable `number1` to pointer variable `fPtr`.
   C. Print the value of the object pointed to by `fPtr`.
   D. Assign the value of the object pointed to by `fPtr` to variable `number2`.
   E. Print the value of `number2`.
   F. Print the address of `number1`. Use the `%p` conversion specifier.
   G. Print the address stored in `fPtr`. Use the `%p` conversion specifier. Is the value printed the same as the address of `number1`?

2. Write the function prototype for a function called `exchange` that takes two pointers to floating-point numbers `x` and `y` as parameters and does not return a value.

3. Find the error in each of the following program segments. If the error can be corrected, explain how.

   A.
   ```
   int *number;
   printf( "%d\n", *number );
   ```

   B.
   ```
   float *realPtr
   long *integerPtr;
   integerPtr = realPtr;
   ```

   C.
   ```
   int * x, y;
   x = y;
   ```

   D.
   ```
   short int *numPtr, result;
   void *genericPtr = numPtr;
   result = *genericPtr + 7;
   ```

   E.
   ```
   float x = 19.34;
   float xPtr = &x;
   printf( "%f\n", xPtr );
   ```

F.
```
char *s;
printf( "%s\n", s );
```

4. Answer each of the following. Assume that single-precision floating-point numbers are stored in 4 bytes, and that the starting address of the array is at location `1002500` 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 10 elements, and initialize the elements to the values `0.0, 1.1, 2.2, …, 9.9`. Assume the symbolic constant `SIZE` has been defined as 10.
   B. Define a pointer, `nPtr`, that points to an object of type `float`.
   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 1 position of precision to the right of the decimal point.
   D. Give two separate statements that assign the starting address of array numbers to the pointer variable `nPtr`.
   E. Print the elements of array `numbers` using pointer/offset notation with the pointer `nPtr`.
   F. Print the elements of array `numbers` using pointer/offset notation with the array name as the pointer.
   G. Print the elements of array `numbers` by subscripting pointer `nPtr`.
   H. Refer to the fifth element 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`.
   I. Assuming that `nPtr` points to the beginning of array `numbers`, what address is referenced by `nPtr + 8`? What value is stored at that location?
   J. Assuming that `nPtr` points to `numbers[5]`, what address is referenced by `nPtr -= 4`. What is the value stored at that location?

5. Suppose that the following declarations are in effect:

```
int a[] = {5, 15, 34, 54, 14, 2, 52, 72};
int *p = &a[1], *q = &a[5];
```

   A. What is the value of `*(p+3)`?
   B. What is the value of `*(q-3)`?
   C. What is the value of `q-p`?
   D. Is the condition `p < q` true or false?
   E. Is the condition `*p < *q` true or false?

   Justify your answer.

6. What will be the contents of the array `a` after the following statements are executed?

```
#define N 10

int a[N] = {1, 2, 3, 4, 5, 6, 7, 8, 9, 10};
int *p = &a[0], *q = &a[N-1], temp;

while (p < q) {
   temp = *p;
   *p++ = *q;
   *q-- = temp;
}
```

7. The following code prints the byte representation of program objects:

```
typedef unsigned char *byte_pointer;

void show_bytes(byte_pointer start, int len) {
   int i;
   for (i = 0; i < len; i++)
      printf(" %.2x", start[i]);
   printf("\n");
}
```

This code uses casting to circumvent the type system. Similar functions are easily defined for other data types. Consider the following three calls to `show_bytes`:

```
int val = 0x87654321;
byte_pointer valp = (byte_pointer) &val;
show_bytes(valp, 1); /* A. */
show_bytes(valp, 2); /* B. */
show_bytes(valp, 3); /* C. */
```

Indicate which of the following values will be printed by each call on a little-endian machine and on a big-endian machine:

A. Little endian: _____ Big endian: _____
B. Little endian: _____ Big endian: _____
C. Little endian: _____ Big endian: _____

8. What would be printed as a result of the following call to `show_bytes`?

```
const char *s = "abcdef";
show_bytes((byte_pointer) s, strlen(s));
```

Note that letters ‘`a`’ through ‘`z`’ have ASCII codes `0x61` through `0x7A`.

9. Consider the following code that attempts to sum the elements of an array `a`, where the number of elements is given by parameter `length`:

```
/* WARNING: This is buggy code */
float sum_elements(float a[], unsigned length) {
   int i;
   float result = 0;

   for (i = 0; i <= length-1; i++)
      result += a[i];
    return result;
}
```

When run with argument `length` equal to 0, this code should return 0.0. Instead it encounters a memory error. Explain why this happens. Show how this code can be corrected.

10. You are given the assignment of writing a function that determines whether one string is longer than another. You decide to make use of the string library function `strlen` having the following declaration:

```
/* Prototype for library function strlen */
size_t strlen(const char *s);
```

Here is your first attempt at the function:

```
/* Determine whether string s is longer than string t */
/* WARNING: This function is buggy */
int strlonger(char *s, char *t) {
   return strlen(s) - strlen(t) > 0;
}
```

When you test this on some sample data, things do not seem to work quite right. You investigate further and determine that data type `size_t` is defined (via `typedef`) in header file `stdio.h` to be `unsigned int`.

A. For what cases will this function produce an incorrect result?
B. Explain how this incorrect result comes about.
C. Show how to fix the code so that it will work reliably.

11. Write a C program which prints

    ```
    I'm on a little-endian machine.
    ```

    or

    ```
    I'm on a big-endian machine.
    ```

    depending on the byte-ordering convention adopted by the machine. Your program should run on any machine, regardless of its word size.

- **Welcome: Sean**
  - LogOut