CS Basics - Exercises
Pointers in C

C. Grothoff and E. Benoist

Fall Term 2018-19

1 Swap

Suppose, that we have two variables `a` and `b`. Write a function `swap` that exchange the values of `a` and `b`. You must pass the arguments by reference, otherwise it won't work.

2 Array = pointer

Suppose that we have the following code

```
int
minimum (int* a, int size);

int
main ()
{
  int array[] = {
    34, 54, 2, 43, 78
  };
  int min = minimum (array, 5);

  printf ("%d\n", min);
  return 0;
}
```

Write the code for the function `minimum()` without using any []. You will use only pointers and the dereferencing operator (∗).

Write another function `minimaxi()` that takes as input an array, and address of two integers. The results (i.e. minimum and maximum of the elements of the array) are written into the integers whose address was passed as arguments.

```
void
minimax (int* array,
         unsigned int length,
          int* min,
          int* max);

int
main ()
{
  int min;
  int max;
  int array[] = {
    34, 54, 2, 43, 78
  };
  minimax (array, 5, &min, &max);
  return 0;
}
```

3 `sizeof()`

Consider the following code:

```
int
main ()
{
  int array[] = {
    34, 54, 2, 43, 78
  };
  int *a = array;

  printf ("%u %u %u %u",
          sizeof (array),
          sizeof (a),
          sizeof (*array),
          sizeof (*a));
  return 0;
}
```

Use it as inspiration to avoid the magic constant "5" in the `minimax()` skeleton.

4 Sorting

You are to implement a simple version of the common `sort` command on UNIX. The `sort` command reads input (from `stdin`), sorts it line-by-line, and writes the sorted output to `stdout`.

You must dynamically allocate an array of strings to call `qsort` for your sorting. If your initial allocation is not large enough, you must detect this and re-allocate a larger array using `realloc()`. Use `getline()` to read input lines of arbitrary length.

Use the `qsort()` function to sort your array. Use `strcmp()` inside your (higher-order) helper function.

Free all allocated memory. Use `valgrind --leak-check=yes` to verify that you did it correctly.

5 Testing

Write a program to generate lines with “random” lines. Use the `random()` function to generate “random” line lengths (say modulo 1024) and “random” printable characters (say A-Za-z).

Test your program by comparing the output of `sort` with your own program’s output using `diff`. Automate the test using a shell script.

6 Bonus: Options

Add support for the options `-f` and `-r` to your program and test scripts:

- Using global variables
- Without using global variables

Also, read up on `getopt()` and use it instead of “manually” parsing options.

7 Bonus: Scoping

Inspect your binary using the `nm -A` and `ldd` tools.

Change the declaration of your helper function to `static`. What does this change in the output of `nm`?

Use `strip` to remove symbols from your object file. What does this change in the output of `nm`?

How big is your binary?