CS Basics - Exercises
Control Structures in C

C. Grothoff and E. Benoist

Fall Term 2017-18

1 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 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.

2 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.

3 Options

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

- Using global variables
- Without using global variables

4 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.

5 Array = pointer

Suppose that we have the following code

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

Write the code for the function `minimum` without using any []. You will use only pointers and 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,int size, int* min, int* max);
main(){
    int min, max;
    int array[]={34,54,2,43,78};
    minimax(array,5,&min,&max);
}
```

6 Implement Bubble Sort on an array of strings

In the file `multidimensionarray2.c` (in the examples for this chapter) write the code for sorting the elements. You need the function to compare strings `strcmp(char*, char*)` that belongs to the standard C library and is declared inside `string.h`. The function `strcmp` returns an integer that is positive if first string is "larger" (i.e. later in the alphabetical order) than the second string.

Example of use

```
void bubbleSort(char** arrayOfStrings,int size);

main(){
  char *arrayOfStrings[] = {"Bonjour", "Gruetzi", "Gruess Euch", ↘
  →"Salut", "Guten Morgen","Bienvenue en cours", "Hello"};
  int size = 7;
  bubbleSort(arrayOfStrings,size);
  int i;
  for(i=0;i<size;i++){
```

```
    puts(arrayOfStrings[i]);
  }
}
```

7 Bonus: make sort interactive

For students having finished the exercise, write an interactive program that does the following:

```
Type the number of strings: 5
Enter string number 1: Hello
Enter string number 2: Bonjour
Enter string number 3: Saali
Enter string number 4: Gruetzi
Enter string number 5: Bienvenue
The sorted strings are:
Bienvenue, Bonjour, Gruetzi, Hello, Saali
```

8 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?