Programming in C and C++
Supervision 1

Andrej Ivašković (ai294)
Compiled on: 9th October 2018

You may email me your work or leave it in my pigeonhole in the Trinity College Great Court mail room.
Please submit the assigned work at least 24 hours before the supervision!

1 Before attempting the problems

This exercise sheet covers the first part of the course, where only core C is considered. The best way to learn about C is by writing C code, and encountering some of its common pitfalls along the way. This sheet contains several programming exercises, but also some bookwork that you need to know (not only for this course, as these ideas will be revisited in *Compiler Construction* and *Concepts in Programming Languages*).

If you don't have any prior C experience, your biggest hurdle might be understanding how pointers work.

When doing the practical exercises, ask yourself the question: "how would I do this in all the other languages I know?"

2 Problems

Some of these questions were taken from the exercises in the old examples sheets. Particularly challenging questions are marked with ⋆.

1. What is the difference between `'a'` and `"a"`?
2. Explain the role of *header files* in C programs.
3. Consider the following code:

```
int main(int argc, char** argv) {
  int a[100], i;
  for (i = 0; i <= 100; i++) {
    a[i] = 5;
  }
  print("ABC");
  return 0;
}
```

Running it results may result in `ABC` never getting printed. Why and when is this the case?

4. (a) Define a simple macro `SWAP(t,x,y)` that naïvely exchanges two arguments of type `t`. Explain why it does not work as expected for `SWAP(int, v[i++], w[f(x)])`. How would you solve this problem?

   (b) Define a macro `SWAP(x,y)` that exchanges two arguments of the same type (for example, `int` or `char`) without using a temporary.

5. Suppose a function is always inlined. Can the compiler safely delete it?

6. Describe the memory layout of a C program in execution.

7. Explain the semantics and usage of `malloc`, `calloc` and `free`.

8. If `p` is a pointer, what does `p[-2]` mean? When is this legal?

9. Look at the C standard library reference and find the signature of `qsort` in `stdlib.h`.

   (a) Explain the meaning of every one of these arguments – in particular, explain how `void` pointers simulate ML-style parametric polymorphism.

   (b) Using the same approach, write implementations of `bubble_sort` and `merge_sort` with the same arguments.

   (c) ⋆ Suggest a way to implement polymorphic `map` in C, recalling that its definition in ML is:

```
1 fun map f [] = []
2   | map f (x::xs) = (f x)::map f xs
```

10. How is a C `struct` represented in memory? How about a `union`? Can changing the order of fields in a `struct` or a `union` decrease the amount of space allocated for it?

Remember to submit your solutions for the labs as well!