UCLA Computer Science 97 Practice Final – Spring 2020

120 points total. Open notes, open computer. Forewarning: **this practice final is very hard** (like the real one will probably be)!! You are **far** from expected to get perfect on it. Don't stress if you don't know everything – use it as a guide to see what topics you need to spend more time on. Answer what you know first before questions you aren't as sure about.

1. (8 mins.) Consider the following pseudocode for an event-based shell program written in JavaScript:

```
class Event { handle() { ... } }
class KeyboardEvent extends Event { handle() { ... } }
class OutputEvent extends Event { handle() { ... } }

function wait_for_event() {
  while (true) {
    if (keyboard_input_waiting) {
      // A keyboard button was pressed by the user
      keyboard_input_waiting = false;
      return KeyboardEvent();
    } else if (output_waiting) {
      // We ran a shell command and have output that needs
      // to be displayed back to the user
      output_waiting = false;
      return OutputEvent();
    }
  }
}

// Main
while (true) {
  const e = wait_for_event();
  e.handle();
}
```

a. (1 min.) What code corresponds to the event loop?

b. (1 min.) What code corresponds to the event handler(s)?

c. (6 min.) Discuss the performance of this particular implementation. Is it a good implementation of event-based programming? Why or why not?

2. (12 min.) In lecture, we talked at a very high-level about threads. A thread is a single sequential flow of instructions within a program. Their main purpose is to allow for concurrency - executing multiple instructions at the same time. Threads share memory, and they can even be running on different CPUs (again, all at the same time).

    a. (2 min.) List **one** main difference between a process and a thread.

b. (10 mins.) The CPython Global Interpreter Lock (GIL) is a mechanism that allows only one thread in a Python program to have control of the interpreter at a time. This means that for any Python program, only **one** of its threads can be in execution at a time. Why, then, would you ever want to multithread a Python process (assuming you're using the CPython interpreter)?

3. (6 min.) Interprocess communication (IPC) refers to the passing of data between different processes. Give **two** examples of IPC that you have used in class, and briefly explain how they work.

4. (10 mins.) In agile development, a *user story* is a simple description of one or more features of a software system, written from the perspective of the end user. Altogether, they describe the complete desired functionality of a software product. Ideally, multiple engineers simultaneously work on implementing different stories in code, and it is easy to figure out which commits correspond to which story. What features of Git would you use to facilitate this?

5. (10 mins.) Most of your final projects used an interpreted language like JavaScript or Python on the back-end. Suppose instead you had used a compiled language like C. What benefits or drawbacks would you encounter?

   (If you *did* use a compiled language on the back-end, suppose you used an interpreted language instead. If you used a serverless database like Firebase, discuss the implications of implementing it using a compiled language like C.)

6. (8 mins.) When the following JavaScript code is run, what does it print out?

```
function callbackHell(a, b) {
  setTimeout(() => {
    console.log("1");
    a(b);
  }, 100000);
  console.log("2");
}

function c(d) {
  setTimeout(() => {
    d();
    console.log("3");
  }, 100000);
  console.log("4");
}

function e() {
  setTimeout(() => {
    console.log("5");
  }, 100000);
  console.log("6");
}

callbackHell(c, e);
```

7. (22 mins.) Consider the following C program:

```c
#include <stdlib.h>

int main() {
  char *buffer = NULL;
  int length = 0;

  int i;
  for (i = 0; i < 10; i++) {
    if (buffer)
      free(buffer);

    buffer = malloc(20);
    length++;
  }

  return 0;
}
```

... with the following Makefile:

```
all: main.c
	gcc -o main -O3 -g main.c
```

   a. (2 min.) List **two** different make commands that can generate the executable `main`.

b. (10 mins.) Consider the following GDB output:

```
(gdb) disas main
Dump of assembler code for function main:
   0x0000000000400490 <+0>:     push   %rbx
   0x0000000000400491 <+1>:     mov    $0xa,%ebx
   0x0000000000400496 <+6>:     mov    $0x14,%edi
   0x000000000040049b <+11>:    callq  0x400480 <malloc@plt>
   0x00000000004004a0 <+16>:    sub    $0x1,%ebx
   0x00000000004004a3 <+19>:    je     0x4004c1 <main+49>
   0x00000000004004a5 <+21>:    test   %rax,%rax
   0x00000000004004a8 <+24>:    je     0x400496 <main+6>
   0x00000000004004aa <+26>:    mov    %rax,%rdi
   0x00000000004004ad <+29>:    callq  0x400450 <free@plt>
   0x00000000004004b2 <+34>:    mov    $0x14,%edi
   0x00000000004004b7 <+39>:    callq  0x400480 <malloc@plt>
   0x00000000004004bc <+44>:    sub    $0x1,%ebx
   0x00000000004004bf <+47>:    jne    0x4004a5 <main+21>
   0x00000000004004c1 <+49>:    xor    %eax,%eax
   0x00000000004004c3 <+51>:    pop    %rbx
   0x00000000004004c4 <+52>:    retq
End of assembler dump.
```

In the generated assembly code for `main`, there isn't a single add instruction (sub means subtract) corresponding to the code "`length++`". How can this be the case?

c. (10 mins.) Consider this truncated output from valgrind:

```
$ valgrind --tool=memcheck ./main
==16102== HEAP SUMMARY:
==16102==     in use at exit: 20 bytes in 1 blocks
==16102==   total heap usage: 10 allocs, 9 frees, 200 bytes allocated
==16102==
==16102== LEAK SUMMARY:
==16102==    definitely lost: 20 bytes in 1 blocks
==16102==    indirectly lost: 0 bytes in 0 blocks
==16102==      possibly lost: 0 bytes in 0 blocks
==16102==    still reachable: 0 bytes in 0 blocks
==16102==         suppressed: 0 bytes in 0 blocks
```

What is the reason for this output? How should the source code be fixed?

8. (6 mins.) Write a `grep` command that would output all preprocessor directives given a C file as input.

9. (8 mins.) Most online Git repositories containing C code contain just that, the code. To run the program, someone that clones the repository needs to compile it themselves. Why is this the case? Why don't developers push the executable as well so that others can skip the compilation step?

10. (8 mins.) Why can't two servers be running on the same port on a computer? What bad things would happen if you could do this?

11. (12 mins.) Suppose you're developing a new language called Viper. Viper is just like Python, except every object now has a free() method that the developer is responsible for calling to deallocate its memory. How is Viper better than Python? How is it worse?

12. (10 min.) Consider the following two C files and their headers:

```c
// parent.c
#include "child.h"
Parent *parent() {
  Parent *p = malloc(sizeof(Parent));
  Child *p = malloc(sizeof(Child));
  p->c = child();
  p->c->p = p;
  return p;
}

int main() { ... }
```

```c
// parent.h
#include "child.h"
struct Parent {
  Child *c;
};
Parent *parent();
```

```c
// child.c
#include "parent.h"
Child *child() {
  Child c = { NULL };
  return c;
}

int main() { ... }
```

```c
// child.h
#include "parent.h"
struct Child {
  Parent *p;
};
Child *child();
```

Assuming all memory is freed properly, what's wrong with the above code? How can it be fixed?