Homework 1 for CS153 (Spring 2016)

Due: on iLearn by Friday 4/15

Instructions:

* Be brief. You will be graded for correctness, not on the length of your answers.
* Make sure to write legibly. Incomprehensible writing will be assumed to be incorrect.

I. Briefly describe how interrupts, faults, and system calls are used in the following three scenarios. Your answer should describe the sequence of execution in these scenarios, and any data structures/state instantiated by the OS during the execution. (6 points)

a) A process does a read system call
b) The disk is done reading data requested by a process
c) A process access an illegal memory address

II. In the state diagram for processes, two edges are missing: there is no edge from Ready to Waiting, and there is no edge from Waiting to Running. Why? (2 points)

III. Consider the following program:

```c
int count = 0;

void twiddledee() {
    int i=0;
    while(i++<2) {count = count + 1;}
}

void twiddledum() {
    int i=0;
    while(i++<2) { count = count * 2;}
}

void main() {
    thread_create(twiddledee); thread_create(twiddledum);
    //wait for both threads to finish
    print count;
}
```

1. Is the variable i shared among the two threads? Is count? (2 points)
2. What is the highest number threads that exist in the program? (2 points)
3. What are the minimum and maximum values of count printed out across all possible executions of this program? (4 points)
2. Describe a potential schedule of execution that will result in the value printed out being equal to 0. Assuming there is only one CPU core, clearly specify when the transitions between the Ready and Running states occurs for each thread in this execution. (6 points)

IV. Consider the following program:

```c
void main() {
    int i = 1;
    int pid = fork();
    printf("i \%d", i);
    if (pid == 0) {
        fork();
        i++;
    }
    else {
        fork();
        exec(some_other_program, arguments);
        fork();
        printf("i \%d", i);
        fork();
    }
}
```

a. How many processes are created during the execution of this program? Explain. (4 points)

b. Assuming the program that is exec’d does not print anything, are the following legal outputs of this program? (yes or no is enough) (4 points)

1. 1 1 2 2 3
2. 1 1 1 2 2
3. 1 2 1 2 1

c. (bonus 1 point) What are the other possible outputs of the program if any?