15-410 Homework Assignment 1. Due Wednesday, February 25, 7pm

1 We Interrupt This Program (10 pts.)

A clever student has designed an “interrupt-safe” pair of `keyboard_handler` and `readchar` functions. The student points out that since there is no state that both functions attempt to modify, nothing can go wrong, even if `keyboard_handler` is invoked by an interrupt while `readchar` is executing.

Assume that all necessary files have been included. Ignore the fact that `lastread` and `lastwrite` might eventually reach `INT_MAX`.

```
#define BUFSIZE 32

static int buf[BUFSIZE];
static int lastread = -1, lastwrite = -1;

void keyboard_handler() {
    int sc;

    sc = inb(KEYBOARD_PORT);

    if ((lastwrite - lastread) < BUFSIZE) {
        lastwrite++;
        buf[lastwrite % BUFSIZE] = sc;
    }

    outb(INT_CTL_REG, INT_CTL_DONE);
}

char readchar() {
    int sc;

    if (lastread < lastwrite){
        lastread++;
        sc = buf[lastread % BUFSIZE];
        return process_scancode(sc);
    }
    else return -1;
}
```

1.1 5 pts

Using the table on this page, describe a scenario in which `readchar` might return an incorrect character value. You should show how the executions of functions `readchar` and `keyboard_handler` are interleaved in time. You must also explain any requirements on the relationship between `lastread` and `lastwrite` for the problem to occur. You are not required to use all spaces in the table.

Execution Trace

time	line number from `readchar`	line number from `keyboard_handler`
0		
1		
2		
3		
4		
5		
6		
7		

1.2 5 pts

Without modifying or moving any existing lines of code, show how to add code to make the `readchar` function safe, and explain why it is safe.

2 Building a Question (10 pts.)

At construction sites, there are many workers doing various jobs to complete their buildings. A construction worker needs a hammer, a hard hat, and nails. These materials are all owned by the construction firm. At this particular site, the foreman is rather relaxed. The hours they work are up to each worker, so they arrive and depart at whatever time suits them. In addition, the construction workers are rather forgetful. They sometimes take equipment home with them and sometimes forget to bring it all back. The foreman has to redistribute equipment in order to keep as many workers as possible busy. The foreman doesn't worry about which individual workers might sit idle.

2.1 5 pts

The foreman proposes implementing an equipment policy using semaphores and has drafted the following code. Nails are not part of the equipment list, due to their plentiful nature. Explain the problem with this policy. You do not need to worry about exact C syntax or invalid parameters.

```
semaphore S_hats;     /* number of spare hats */
semaphore S_hammers;  /* number of spare hammers */

/*
 * This function will be called before any worker arrives.
 * The parameters represent how much equipment the foreman
 * has at the start of the day.
 */

void init(int hard_hats, int hammers){
   semaphore_init(S_hats,hard_hats);
   semaphore_init(S_hammers,hammers);
}

/*
 * Workers call this function when they arrive.
 * The parameters are set to 1 if the worker brings this
 * piece of equipment, and 0 otherwise.
 * This should return when the worker has both a hat
 * and a hammer.
 */

void arrivewith(int hh, int hammer){
  if (hh == 0){
    wait(S_hats);
  }
  if (hammer == 0){
    wait(S_hammers);
  }
}
```

```
/*
 * Workers call this function when they depart.
 * The parameters are what the worker is leaving with
 * the foreman.
 */
void departleaving(int hh, int hammer){
  if (hh == 1){
    signal(S_hats);
  }
  if (hammer == 1){
    signal(S_hammers);
  }
}
```

2.2 5pts

For the second part of this problem, you should supply new code for the function `arrive` to implement an efficient work policy.

```
/*
 * Workers call this function when they arrive.
 * The parameters are set to 1 if the worker brings this
 * piece of equipment, and 0 otherwise.
 * This should return when the worker has both a hat
 * and a hammer.
 */

void arrivewith(int hh, int hammer){









}
```

3 Are we dead yet? (12 pts.)

Deadlocks are rather bad and may occur in even simple situations. Learning to characterize when a system is in a "safe" state and when there is a potential for deadlock is important. In this problem we will examine a system in which each process has provided the operating system with a list of the resources that it will need during execution. In the first part of the problem, the resources are simply listed. In the second part, each process provides the sequence in which it will acquire and release the resources. There are three resources, R, S, and T, and three processes, A, B, and C. For the following situations characterize them as safe (if the resource manager can allocate, in some order, all needed resources to each process and still avoid deadlock), unsafe (not safe), or deadlocked (and hence unsafe), and explain why. To answer the "Why?" question in each part, you may need to draw resource allocation graphs, present safe sequences, and / or make brief proof-like arguments. Please make sure you are concise yet convincing. Whether it's a good idea or not, the resource manager grants all requests that don't violate mutual exclusion.

Suppose that each process indicates which resources it is going to use (as shown in the table below), but does not specify any particular order in which it will use the resources.

Process Traces

Process A	Process B	Process C
R	T	T
S	S	R

3.1 2 pts

Suppose the resource manager allocates S to process A. Is the system in a safe, unsafe, or deadlocked state? Why?

3.2 2 pts

Suppose the resource manager then allocates T to process B, while A continues to hold resource S. Is the system in a safe, unsafe, or deadlocked state? Why?

3.3 2 pts

Suppose that the resource manager allocates R to process C, while A and B hold S and T. Is the system in a safe, unsafe, or deadlocked state? Why?

Now suppose that initially each process is required to completely specify the order in which it will request and release resources, and they provide the information below.

Process Traces

Process A
request(S)
request(R)
release(R)
release(S)

Process B
request(T)
request(S)
release(S)
release(R)

Process C
request(R)
request(T)
release(R)
release(T)

Execution Trace:

Process A	Process B	Process C
request(S)		
	request(T)	

3.4 2 pts

If, after the execution trace shown above, the next event that happens is process C: request(R), which is granted. Is the system safe, unsafe or deadlocked? Why?

3.5 2 pts

Assume that, instead of process C: request(R), the third event is process A: request(R). Is the system be safe, unsafe or deadlocked? Why?

3.6 2 pts

Finally assume that, instead of process C: request(R), the third event is process B: request(S). Is the system be safe, unsafe, or deadlocked? Why?

4 The Main Thing (12 pts.)

When a Unix process begins execution, `main()` is invoked with two parameters, namely an integer specifying the number of "command line" word strings and the word string vector. By convention, the zero'th string is the name of the program.

4.1 3 pts

Where in the process's address space are the string count, the string vector, and the strings themselves stored? Draw a picture of how they are laid out.

4.2 3 pts

Could some of those parts be stored somewhere else? If so, which and where?

4.3 6 pts

In your opinion, which place is better, and why?