Homework 6

Problem 1

```
#include <stdio.h>
#include <string.h>
#include <stdlib.h>
#include <signal.h>
#include <setjmp.h>

pid_t pid;
int counter = 2;

jmp_buf jbuf;

void handler1(int sig)
{
    counter = counter - 1;
    printf("%d", counter);
    fflush(stdout);
    longjmp(jbuf, 4);
}

int main()
{
    signal(SIGUSR1, handler1);
    printf("%d", counter);
    fflush(stdout);

    int rc = setjmp(jbuf);
    if (rc == 0) {

        if ((pid = fork()) == 0) {
            while(1) {};
        }
        kill(pid, SIGUSR1);
```

```
        waitpid(-1, NULL, 0);
        counter = counter + 1;

    } else {
        printf("%d", rc);
        fflush(stdout);
    }
    printf("%d\n", counter);
    exit(0);
}
```

What's the output of the given program?

Problem 2

```
#include <signal.h>
#include <stdio.h>
#include <stdlib.h>

void handler(int sig)
{
    static int beeps = 0;
    printf("BEEP\n");
    if (++beeps < 3) {
        fork();
        alarm(1); /* next SIGALRM will be delivered in 1s */
    } else {
        printf("BOOM!\n");
        exit(0);
    }
}

int main()
{
    signal(SIGALRM, handler); /* install SIGALRM handler */
```

```
    alarm(1); /* next SIGALRM will be delivered in 1s */
    /* signal handler returns control here each time */
    while (1);
    exit(0);
}
```

1. How many BEEPs will be printed if you run the above program?
2. How about BOOMs?

Problem 3

```
#include <stdio.h>
#include <string.h>
#include <stdlib.h>
#include <unistd.h>

#define MAXLEN 1000
int main()
{
    char *bin = "ls";
    char *argv[] = { bin, NULL };
    char buf[MAXLEN];
    /* Fetch the environment variable 'PATH' into 'env_path' */
    char *env_path = __(1)__
    strcpy(buf, env_path);
    char *p = buf;
    char *delim;
    char path[MAXLEN];
    /* Paths are separated by colon ':' in 'PATH'. */
    while ( __(2)__ ) {
        *delim = '\0';
        strcpy(path, p);
```

```
        /* Suppose now ‘path’ is ‘/usr/bin’,
         * how do we obtain ‘/usr/bin/ls’ in ‘path’?
         * Hint: use ‘strcat’. */
        __(3)__
        /* Execute the command in ‘path’ */
        __(4)__
        /* Make ‘p’ point to the next path string */
        __(5)__
    }
    fprintf(stderr, "No binary named %s found\n", bin);
    exit(1);
}
```

Questions:

1. Fill in the blanks above.
2. What does the program above do? Try it on your Linux!
3. Is it necessary to check the return value of ‘execve’? Why?
4. What does it mean if ‘execve’ returns an error?