CS330: Operating Systems
Quiz#1

Name:
Roll No.:

1. Consider the following program.

```
#include <stdio.h>
#include <unistd.h>
#include <sys/types.h>
#include <sys/wait.h>
#include <stdlib.h>

typedef struct {
   int x;
} global_t;

global_t *g;

int voodooFactorial (int n)
{
        if (n==1) return 1;
        if (fork() == 0) g->x++;
        else wait(NULL);
        return n*voodooFactorial(n-1);
}

int main(void)
{
        int y;
        g = (global_t*)malloc(sizeof(global_t));
        g->x = 5;
        y = voodooFactorial(g->x-2);
        printf("%d\n", y*g->x);
        return 0;
}
```

What are the possible outputs of this program? **(6 points)**

Solution: 42 36 36 30. Key observations: First, there will be four processes because `fork` will be called thrice. Second, all four processes will return six to `y` because even if some processes start off in the middle of the recursion, the stack is copied from the parent during `fork`.

Grading policy: 1.5 point for each correct answer appearing in the correct order.

2. Consider the following programs. Assume that `execv` does not encounter any error during execution.

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

int main(void)
{
        int x = getpid();
        FILE *fp = fopen("pid.txt", "w");
        fprintf(fp, "%d\n", x+3);
        fclose(fp);
        execv("reader",NULL);
        if (fork() == 0) printf("%d\n", x-getppid());
        return 0;
}
```

The reader.c program is shown below.
This program is compiled into the executable named reader.

```
#include <stdio.h>
#include <sys/types.h>
#include <unistd.h>

int main(void)
{
        FILE *fp = fopen("pid.txt", "r");
        int x;
        fscanf(fp, "%d", &x);
        fclose(fp);
        printf("%d\n", getpid()-x);
        return 0;
}
```

What are the possible outputs? **(2 points)**

Solution: -3. Key observations: First, `execv` does not create a new process and hence, preserves the pid. Second, successful execution of `execv` does not return to the calling program.

Grading policy: 2 points for correct answer. Zero otherwise. If you write the correct answer and the output of the `printf` after `execv`, you will lose one point.

3. A program executes eight system calls. Four of these calls are `fork` calls and three of the remaining calls require disk I/O. The last system call is the `exit` call, through which the program terminates. Write down the number of mode switches and context switches experienced by the parent process only. **(2 points)**

Solution: Each system call, except `exit`, requires two mode switches. The `exit` call requires only one mode switch because the calling process is terminated while in the kernel mode. Only the three disk I/O calls require context switches. So, there are **15 mode switches** and **3 context switches**.

Grading policy: One point each for correct counts of mode switches and context switches. I have also given one point for six context switches, although this answer is wrong because a context switch involves saving the state of current process and restoring the state of the next process. This is not equivalent to two context switches.