Homework 13

Problem 1

```
1     #include “csapp.h”
2     void *thread(void *vargp);
3
4     int main()
5     {
6         pthread_t tid;
7
8         Pthread_create(&tid, NULL, thread, NULL);
9         exit(0);
10    }
11
12    /* thread routine */
13    void *thread(void *vargp)
14    {
15        Sleep(1);
16        printf(“Hello, world!\n”);
17        return NULL;
18    }
```

The program above is expected to output “Hello, world!” after sleeping for 1 second. However, nothing is printed while it is running. Describe the reason and modify the main routine to fix the bug.

The problem is that the main thread calls exit without waiting for the peer thread to terminate. The exit call terminates the entire process, including any threads that happen to be running. So the peer thread is being killed before it has a chance to print its output string.

We can fix the bug by replacing the exit function with either Pthread_exit, which waits for outstanding threads to terminate before it terminates the process, or Pthread_join which explicitly reaps the peer thread.

Problem 2

Is there potential deadlock in the following program? Describe the reason briefly.

```
Initially: a = 1, b = 1, c = 1;

Thread 1:          Thread 2:          Thread 3
P(a);              P(c);              P(c);
```

```
P(b);                P(b);                V(c);
V(b);                V(b);                P(b);
P(c);                V(c);                P(a);
V(c);                P(a);                V(a);
V(a);                V(a);                V(b);
```

Intuitively, one possible result is that Thread 1 holds a, Thread 2 holds c and Thread 3 holds b, while they request c, b and a respectively. The deadlock occurs.

Formally, Thread 1 may hold (a, b) or (a, c) simultaneously, Thread 2 may hold (c, b) simultaneously and Thread 3 may hold (b, a) simultaneously. There exists loop in these tuples, which causes the deadlock.

Problem 3

Recall the `Tfgets` function mentioned in Homework 9, which uses signal and non-local jump to implement a timeout version `Fgets`. Now try to implement the same function with I/O multiplex. Hint: the last parameter of `select()` can be used to indicate timeout. You can look up Linux Programmer's Manual for detail. (In Linux, "man select")

```
1     #include "csapp.h"
2
3     #define TIMEOUT 5
4
5     char *tfgets(char *s, int size, FILE *stream)
6     {
7         struct timeval tv;
8         fd_set rfds;
9         int retval;
10
11        FD_ZERO(&rfds);
12        FD_SET(0, &rfds);
13
14        /* Wait for 5 seconds for stdin to be ready */
15        tv.tv_sec = 5;
16        tv.tv_usec = 0;
17        retval = select(1, &rfds, NULL, NULL, &tv);
18        if (retval)
19            return fgets(s, size, stream);
20        else
21            return NULL;
```

Problem 4

Determine the number of page table entries (PTEs) that are needed for the following combinations of virtual address size (n-bit) and page size (P):

n	$P=2^p$	# PTE
16	2K	32
16	4K	16
32	4K	1M
32	8K	512K