Problem 1. (50 points)
Show what this program prints and explain it in detail. Do not compile and run this program.

```
pid_t pid;
void handler1(int sig) {
     printf("ONE ");
     fflush(stdout);
     kill(pid, SIGUSR1);
}
void handler2(int sig) {
     printf("THREE ");
     exit(0);
}

int main() {
signal(SIGUSR1, handler1);
 if ((pid = fork()) == 0) {
     signal(SIGUSR1, handler2);
     kill(getppid(), SIGUSR1);
     while(1) ;
} else {
pid_t p;
int status;
if ((p = wait(&status)) > 0) printf("TWO ");
}
}
```

Output:
ONE THREE TWO

The program first sets signal handler1 in the parent process for the SIGUSR1 signal.
In the child process. It then creates a child process.

Within the child process, a second handler "handler2" is then set up for the SIGUSR1 signal and then will send SIGUSR1 to the parent process. Once the parent receives SIGUSR1, it will print "ONE" from within the handler, flush the stdout buffer, and send SIGUSR1 to the child process. Now, the parent process is waiting for the child to change status. The child process though is in an infinite loop for the time being. It can only be terminated by a signal.

Back within the child process, when the child receives SIGUSR1, it invokes handler 2, prints "THREE" and then the handler terminates the child process. The if condition at the end of the parent is true since the process in the child has terminated. "TWO" is then printed and then the program terminates

Problem 2. (50 points)
Do not compile and run these programs. You should answer all questions by reviewing the code.
Part 1 (25 out of 50 points)

What does this program print? Will it always print the same text? Explain in detail why or why not.

```
#include "csapp.h"
#define N 2

void *thread(void *vargp);
char **ptr;
int cnt;

int main() {
     int i;
     pthread_t tid;
     pthread_t tid[N];
     char *msgs[N] = { "Hello from zero", "Hello from one" };
     ptr = msgs;
     cnt = 0;
     for (i = 0; i < N; i++)
          Pthread_create(&tid[i], NULL, thread, (void *) &i);
      for (i = 0; i < N; i++)
          Pthread_join(tid[i], NULL);
     Pthread_exit(NULL);
     }

void *thread(void *vargp) {

/* The modulo operator, %, produces the remainder of an integer
division. */
     int myid = (int)vargp;
      if ((myid % 2) == 1) cnt++;
     else cnt--;
     printf("[%d]: %s (cnt=%d)\n", myid, ptr[myid], cnt);
     return NULL;
}
```

This program will not print anything and will terminate due to an error. One is due to declaring a pthread called tid and the other is an array of the same name. This is not allowed because only one array should be used to store the indexes to the threads.

There are other issues that would lead to undefined behavior in the output. One is that the memory address of the loop variable i, is passed directly into the thread. This value could change before the thread accesses it and dereferences the pointer, which leads to a problem where the threads access the same data as another thread or even an out of bounds index. Another issue is that cnt is not protected by a mutex lock. This is important because cnt is a global variable. When a thread accesses it, it would not be in sync due to the first issue where threads could be sharing the same data. The final issue is that when the pointer to vargp is passed into the thread, it is converted directly to an int. However, because the value passed into the thread is the memory address of i, the code attempts to convert the actual memory address to an integer when the pointer to the index should just be dereferenced for each thread. This issue would be prone to segmentation faults since the thread attempts to access an index that has not been allocated.

Part 2 (25 out of 50 points)
The purpose of the following program is to create four threads, indexed as 0, 1, 2, 3, and have each thread print its own index. Explain what this program prints. Will it always print the same text? Explain in detail why or why not. The use of variable ptr has a very specific purpose. What problem is the programmer trying to solve by using variable ptr? Has he succeeded? Explain in detail why or why not.

```
#include "csapp.h"
#define N 4
void *thread(void *vargp);
int main() {
      pthread_t tid[N];
      int i, *ptr;
      for (i = 0; i < N; i++) {
           ptr = Malloc(sizeof(int));
          *ptr = i;
          Pthread_create(&tid[i], NULL, thread, ptr); }

      for (i = 0; i < N; i++)
          Pthread_join(tid[i], NULL);
        exit(0);
      }

/* thread routine */
void *thread(void *vargp) {
      int myid = *((int *)vargp);
      Free(vargp);
      printf("Hello from thread %d\n", myid);
      return NULL;
 }
```

Example Output:

```
Hello from thread 0
Hello from thread 2
Hello from thread 3
Hello from thread 1
```

The program will not always print the text in the same order though since the program creates threads. Threads may have different schedules on each run and those schedules are dependent on the operating system's instructions. The programmer is attempting to create 4 separate threads and return their indexes when they finish.

By using the ptr variable, the programmer is making sure that each thread index is allocated to a unique memory address. This in turn ensures that each thread has its own index and that each thread is accessing its own data. Each thread would just need to free the pointer to the index after it has been assigned

If the ptr variable was not there all threads would be accessing and modifying the same information concurrently leading to incorrect output where the same index could be printed multiple times. It also prevents other bugs like race conditions. So, the programmer has succeeded with his implementation.