Name: UID#: DirectoryID:

CMSC216: Practice Final Exam B

Spring 2025
University of Maryland

Exam period: 20 minutes
Points available: 40

Problem 1 (10 pts): Examine the code to the right and describe what you expect its output to be. Explain why or why not you would expect to see any specific ordering in the output of the program.

```c
#include <stdio.h>
#include <unistd.h>
#include <stdlib.h>
#include <wait.h>

int main(){
  for(int i=0; i<5; i++){
    pid_t p = fork();
    if(p  != 0){
      wait(NULL);
      printf("iter %d, %d from %d\n",
             i,getpid(),getppid());
      fflush(stdout); // output now
      break;          // quit loop
    }
  }
  exit(0);
}
```

Problem 2 (10 pts): Nearby is the output of `pmap` showing page table virtual memory mapping information for a running program called `memory_parts`. Answer the following questions about this output.

(A) Certain addresses of memory are marked with the annotation `r-x`. Explain what this means and what kind of information you would expect to find in those addresses.

(B) Why does `pmap` only show a limited number of virtual addresses? What would happen if the program attempted to access an address not listed in the output? Example: address `0x00` is not in the listing.

```
> pmap 7986
7986:   ./memory_parts
00005579a4abd000      4K r-x-- memory_parts
00005579a4cbd000      4K r---- memory_parts
00005579a4cbe000      4K rw--- memory_parts
00005579a4cbf000      4K rw---   [ anon ]
00005579a53aa000    132K rw---   [ heap ]
00007f441f2e1000   1720K r-x-- libc-2.26.so
00007f441f48f000   2044K ----- libc-2.26.so
00007f441f68e000     16K r---- libc-2.26.so
00007f441f692000      8K rw--- libc-2.26.so
00007f441f694000     16K rw---   [ anon ]
00007f441f698000    148K r-x-- ld-2.26.so
00007f441f88f000      8K rw---   [ anon ]
00007f441f8bb000      4K r---- gettysburg.txt
00007f441f8bc000      4K r---- ld-2.26.so
00007f441f8bd000      4K rw--- ld-2.26.so
00007f441f8be000      4K rw---   [ anon ]
00007fff96ae1000    132K rw---   [ stack ]
00007fff96b48000     12K r----   [ anon ]
00007fff96b4b000      8K r-x--   [ anon ]
 total            4276K
```

Problem 3 (10 pts): Nearby is a matrix/vector function which performs poorly. Create a new version of this function that **optimizes the memory access pattern**. Show your code and give a brief description of why the changes you made should improve performance.

```
int subcol_BASE(matrix_t mat, vector_t vec) {
  for(int j=0; j<mat.cols; j++){
    for(int i=0; i<mat.rows; i++){
      int elij = MGET(mat,i,j);
      int veci = VGET(vec,i);
      elij -= veci;
      MSET(mat,i,j,elij);
    }
  }
  return 0;
}
```

```
int subcol_OPT(matrix_t mat, vector_t vec) {
// YOUR CODE HERE
```

WHY CHANGES IMPROVE PERFORMANCE:

Problem 4 (5 pts): To further optimize the `subcol_opt()` function, a common strategy is to utilize multiple threads. Describe briefly how this might be done. Include in your answer.

(A) How the work to be done is divided among threads

(B) How changes to shared data will be coordinated to ensure safety.

Problem 5 (5 pts): Consider the code sample nearby which prints logging messages to either the screen or a log file as dictated by the `USE_LOGFILE` variable. Describe how one could eliminate the conditional `if/else` and all the `fprintf()` calls using **I/O redirection system calls** within the program.

```
{
  if(USE_LOGFILE){
    fprintf(logfile,"Updating DB\n");
  }
  else{
    printf("Updating DB\n");
  }
  update_db();
  if(USE_LOGFILE){
    fprintf(logfile,"Syncing files\n");
  }
  else{
    printf("Syncing files\n");
  }
  file_sync();
  ...;
}
```