Name: _____

Write your name and computing ID above. Write your computing ID at the top of each page in case pages get separated. Sign the honor pledge below.

Generally, we will not answer questions about the exam during the exam time. If you think a question is unclear and requires additional information to answer, please explain how in your answer. For multiple choice questions, write a * next to the relevant option(s) along with your explanation.

On my honor as a student I have neither given nor received aid on this exam.

_____

1. Suppose there is a directory `A` with the following access control list

```
user:foo:r-x
user:bar:rwx
group:quux:r-x
other:--x
```

And inside that directory there is a file `B` with the following access control list:

```
user:bar:rw-
group:quux:r--
other:---
```

(a) (4 points) Which of the following is true about how A and B can be accessed? **Select all that apply.**

- ☐ a program running as user `foo`, group `xyxxz` can list the directory `A` and see that it contains `B`
- ☐ a program running as user `delta`, group `quux` can read the contents of `B`
- ☐ a program running as user `bar`, group `quux` can modify the contents of `B`
- ☐ a program running as user `bar`, group `xyxxz` can modify the contents of `B`

(b) (4 points) Would it be possible to achieve the same effect as the above access control lists using chmod-style permissions? Explain **briefly**.

2. Suppose several (single-threaded) processes with PIDs 1999, 2000, and 2001 are all running the same program with the same user ID. As part of its startup, the program sets up the following `handle_sigusr1` function as a signal handler for SIGUSR1, and `handle_sigusr2` as a handler for SIGUSR2:

```
void handle_sigusr1(int signum) {
    kill(2000, SIGUSR2);
    kill(2001, SIGUSR2);
}

void handle_sigusr2(int signum) {
    if (getpid() == 2000)
        write(STDOUT_FILENO, "X", 1);
    if (getpid() == 2001)
        write(STDOUT_FILENO, "Y", 1);
    write(STDOUT_FILENO, "Z", 1);
}
```

Each of the processes is initially running the following loop:

```
char buffer[100];
do {
    fgets(buffer, sizeof buffer, stdin);
} while (strcmp(buffer, "QUIT\n") == 0);
```

For the following questions assume kill and write do not fail and the processes do not exit the loop.

(a) (4 points) Suppose SIGUSR2 is received by pid 1999, starting its signal handler. Which of the following is true about this situation? **Select all that apply.**

- ☐ a call to `strcmp` may have started in pid 1999 and not finished
- ☐ a call to `fgets` may have started in pid 1999 and not finished
- ☐ SIGUSR2 will be masked in pid 2000 and pid 2001
- ☐ the value of `buffer` for pid 1999 may be stored in the OS's memory temporarily while the signal handler runs

(b) (5 points) Suppose SIGUSR1 is sent to pid 1999, starting its signal handler. Which of the following is true about this situation? **Select all that apply.**

- ☐ at least one SIGUSR2 handler will run before pid 1999's SIGUSR1 handler returns
- ☐ at least two context switches will occur before pid 1999's signal handler returns
- ☐ pid 1999 will make multiple system calls before its signal handler returns
- ☐ pid 2001's SIGUSR2 signal handler might run before pid 2000's
- ☐ more than three context switches could occur before pid 1999's signal handler returns

3. Consider this code:

```
pid_t pids[103] = {0};

int fd = open("output.txt", O_WRONLY | O_CREAT | O_TRUNC);
dup2(STDOUT_FILENO, 102);

for (int i=100; i<103; i++) {
  pids[i] = fork();
  if (pids[i] == 0) {
    dup2(fd, i);
    write(i, "foo", 3);
    exit(0);
  } else {
    write(i, "p1", 2);
  }
}

for (int i=100; i<103; i++) {
  waitpid(pids[i], NULL, 0);
}

write(100, "p2", 2);
write(101, "p2", 2);
write(102, "p2", 2);

return 0;
```

For the following questions, assume that `fork` and `waitpid` and `open` do not fail, and the `open` call returns `3`.

(a) (4 points) What will be written to stdout?

- ◯ `foofoofoo`
- ◯ `p1p1p1foofoofoop2p2p2`
- ◯ `p1p1p1p2p2p2p2`
- ◯ `p1p2`
- ◯ `p2p1`
- ◯ (nothing)

(b) (4 points) What strings will appear in `output.txt`? **Select all that apply.**

- ☐ `foo`
- ☐ `p1`
- ☐ `p2`

(c) (4 points) How many *new* processes are created by this code?

_____

4. Consider a system with:
   - single-level page tables, with 4-byte page table entries
   - 65536 byte (2^{16} byte) pages
   - 30-bit virtual addresses
   - 28-bit physical addresses

(a) (8 points) Fill in the blanks. You may leave your answers as unsimplified arithmetic expressions.

Suppose the page table base register is set to physical address `0x40000`. To accessing virtual address `0x89ABC` on behalf of a program, the processor will access a page table entry at physical address _____. If that page table entry is valid and its permission bits match the corresponding access and contains physical page number 0x3, then the processor will access physical address _____.

(b) (4 points) Suppose a process running on this system has the following memory layout:
   - `0x000000-0x0FFFFF`: invalid
   - `0x100000-0x2FFFFF`: program code and data
   - `0x300000-0x3FFFFF`: invalid
   - `0x400000-0x42FFFF`: stack
   - `0x430000-0x6FFFFF`: invalid
   - `0x700000-0x7FFFFF`: OS data, only accessible in kernel mode, and loaded in advance
   - `0x800000-` maximum possible: invalid

Suppose, when the program initially starts the OS only sets up the page table entries for OS data. Then, as the program runs, it allocates memory and sets up other page table entries on demand, one at a time, in response to page faults (exceptions triggered when code tries to access an invalid page).

If the program starts and accesses code at addresses `0x138F00--0x1472FF`, and its stack at address `0x42FF00--0x42FFF8`. After this happens, how many physical pages will be allocated for the process (excluding allocations for OS data)?

5. Suppose a single-core system is running two processes A and B. Initially process A is running in user mode on the processor. Then, the following sequence of events happens:
   1. Process A starts reading from a file on disk
   2. The operating system requests the disk device access that file's data
   3. While the disk is accessing the file, Process B starts running in user mode
   4. Process B performs some computations
   5. The disk completes access to the file's data, and the operating system retrieves that data
   6. Process B requests to display the result of that computation to the screen
   7. The result is displayed to the screen.
   8. Process A resumes running and processes the data retrieved from the file
   9. When Process A tries to write to a data structure on the stack, the operating system allocates an additional physical page for its stack

(a) (4 points) When will at least one system call to be started? Write the numbers of the steps listed above.

_____

(b) (4 points) When will an exception triggered by an event external to the running program occur? Write the numbers of the steps listed above.

_____

(c) (4 points) Will any exception not described by the previous two questions occur? If so, identify it briefly.

6. (10 points) Suppose we have an library `libfoobar.a` and executable `foobar` that is written in a combination of C, assembly and FORTRAN. To compile the FORTRAN code, a FORTRAN file `name.f` is converted to a C file `name.c` using the '`f2c`' utility. The source files for the library are `futils.f`, `asmutils.s`, and `cutils.c`. The executable is built using the library and the source file `main.c`, and the system libraries `-lm` and `-lf2c`. Overall both the executable and library can be built using the following sequence of commands:

```
clang -c main.c
f2c futils.f
clang -c futils.c
clang -c asmutils.s
clang -c cutils.c
ar rcs libfoobar.a futils.o asmutils.o cutils.o
clang -o foobar main.o -L. -lfoobar -lm -lf2c
```

Fill in the blanks in the following Makefile to automate this process. You may not need all lines. You may refer to the reference sheet for information about selected `make` syntax.

```
all: foobar libfoobar.a

%.o: %.c

        ------------------------------------------------------


_________________: futils.f

        ------------------------------------------------------

        ------------------------------------------------------

asmutils.o: asmutils.s
        clang -c asmutils.s

libfoobar.a: futils.o asmutils.o cutils.o
        ar rcs libfoobar.a futils.o asmutils.o cutils.o

foobar: main.o ________________________________________
        clang -o foobar main.o -L. -lfoobar -lm -lf2c
```

7. (6 points) Suppose a system has a 40-bit addresses and 2^{16} byte direct-mapped cache with 2^7-byte blocks.

   Suppose the cache currently stores a value retrieved from memory address 0x12345678. Accessing some other address would cause this byte to be replaced by new data in the cache. Give an example of such an address.

8. (8 points) Suppose a system has a four-level page table with 8192 byte-pages and 8-byte page table entries. As a result each page table has 1024 (2^{10}) page table entries.

   To make allocating space for page tables simpler, the system chooses its address sizes such that each page table is one page in size.

   Consider the following correspondance of virtual pages to physical pages. In the table below, virtual page numbers are split up to support page table lookups. 'VPN part 1' represents the part of the virtual page number used as the index into first level table; 'VPN part 2' into the sceond level table; and so on.

VPN part 1	VPN part 2	VPN part 3	VPN part 4	physical page number
0	0	1	4	100
0	0	1	5	101
0	0	1	6	102
0	9	1	4	103
17	0	0	0	400

   To represent the above table in the page table data structure, how many page tables will be required at each level? Complete the following table.

first level tables	
second level tables	
third level tables	
fourth level tables	