CS 31 Homework 6: Processes (due December 5, 2023 at the start of class)

Names of **all** students who worked on this submission::

Question 1

For the code snippet shown below (assume that all the calls to `fork()` succeed), answer the following questions:

A. Draw a process hierarchy diagram that results from the execution of the code shown below. Your diagram should be similar to Figure 2 in Section 13.2 of the textbook, where you draw a node for every process and arrows from parent to child processes.

- Label each node with a letter to indicate the order in which it was spawned. In cases where the order is not determined, choose a possible order.
- Next to each node, write the output value(s) that the process prints out with `printf()`.

B. After this code executes, are there any zombie processes? Explain your answer in a sentence or two.

```
int i = 0;
pid_t pid;

printf("%d ", i);
for(i = 1; i < 3; i++) {
   pid = fork();
   printf("%d ", i);
}

if(pid != 0) {
  wait(NULL);
} else {
  exit(0);
}
```

Question 2

Consider the code snippet shown below (and assume all calls to `fork()` succeed).

A. Draw the execution timeline corresponding to the code's execution showing a possible ordering of `fork()` and `wait()` calls from the proceses involved. Use Figure 7 in Section 13.2 of the textbook as an example. Note: There are no newlines in the `printf` calls below, so the output should all be on a single line.

```
pid_t pid1, pid2;

printf("1 ");
pid1 = fork();
if (pid1 == 0) {
  pid2 = fork();
  printf("2 ");
  if (pid2 == 0) {
    printf("3 ");
    exit(0);
  } else {
    printf("4 ");
    wait(NULL);
    printf("5 ");
    exit(0);
  }
} else {
  printf("6 ");
  wait(NULL);
  printf("7 ");
}
```

B. Which of the following outputs below are possible from executing the above code? For any that are not, describe in one sentence why not.

i) `1 6 2 3 2 4 7 5`

ii) `1 2 2 4 5 3 6 7`

iii) `1 2 2 3 4 5 6 7`

iv) `1 6 2 4 2 3 5 7`