CSE3100 – Systems Programming
Practice Exam 2

Problem 1. pipe-sort.
In this problem, we use two child processes and pipes to sort an array of integers in the parent process into increasing order.

Specifically, the parent process generates `n` random integers, where `n` must be a positive even number, and saves them in array `u`. Then it creates pipes and forks two child processes so that child process 1 sorts the first half of array `u` and child process 2 sorts the second half. Both child processes just call `qsort()` to sort their array in place. Then they send the sorted integers to the parent process through pipes.

The parent process receives the sorted integers from both child processes through pipes, and saves them into array `u`. Then it merges the two sorted halves into array `sorted`.

The child processes call `qsort()` to sort arrays and the compare function for integers is provided in `compare_int()`.

Tasks.

The starter code has four files: `pipe-sort.c`, `pipe-sort.h`, `main.c`, and a `Makefile`. We will mainly work on the `pipe_sort()` function in the `pipe-sort.c` file, but may add a few helper functions if needed. Search `TODO` to find the location. Read the comments and code carefully.

When we run the program `pipe-sort`, the number of integers `n` must be specified. Optionally, `-s` option specifies a seed, `-p` option specifies the number of integers to be printed (which is useful when `n` is large), `-u` option makes the program print unsorted array and exit. For example, the following session sorts 10 integers, generated from the default seed 3100, and only prints first 3 integers in the sorted array.

```
$ ./pipe-sort 10 -p3
seed=3100 n=10 print_sorted=1 num_printed=3
0
0
2
```