Problem 4 of 8

Processes

A child process inherits all signal handlers of its parent. You may assume system calls will not return any errors. Each call to printf has an associated call to fflush.

```
#include <unistd.h>
#include <stdlib.h>
#include <signal.h>
#include <stdio.h>

int areCute = 0;

void sigchld_handler(int sig) {
  waitpid(-1, NULL, 0)
  if (areCute)
    Sio_puts("are cute! ");
  else
    Sio_puts("are kinda terrifying! ");
}

int main() {
  sigset_t mask;

  Sigemptyset(&mask);
  Sigfillset(&mask);
  Signal(SIGCHLD, sigchld_handler);

  if (fork() == 0) {
    printf("Red Pandas ");
    Sigprocmask(SIG_BLOCK, &mask, NULL);
    areCute = 1;
    Sigprocmask(SIG_UNBLOCK, &mask, NULL);
    exit(0);
  } else {
    printf("Cockroaches ");
    Sigprocmask(SIG_BLOCK, &mask, NULL);
    areCute = 1;
    Sigprocmask(SIG_UNBLOCK, &mask, NULL);
  }
  return 0;
}
```

Select the possible things printed.

Red Pandas Cockroaches are cute! are kinda terrifying!

◯ Y ◯ N

Cockroaches Red Pandas are kinda terrifying!

◯ Y ◯ N

Red Pandas Cockroaches are cute!

◯ Y ◯ N

Cockroaches Red Pandas are cute!

◯ Y ◯ N

Cockroaches Red Pandas are kinda terrifying! are cute!

◯ Y ◯ N

Red Pandas Cockroaches are kinda terrifying!

◯ Y ◯ N

Cockroaches Red Pandas are cute! are kinda terrifying!

◯ Y ◯ N

Cockroaches are kinda terrifying! Red Pandas are cute!

◯ Y ◯ N

Red Pandas are cute! Cockroaches are kinda terrifying!

◯ Y ◯ N

Red Pandas are kinda terrifying! Cockroaches are cute!

◯ Y ◯ N

Consider the C program below.

```
int main() {
    if (fork() == 0) {
        if (fork() == 0) {
            wait(NULL);
            printf("a");
        } else {
            printf("b");
            wait(NULL);
        }
    } else {
        wait(NULL);
        if (fork() == 0) {
            printf("c");
        } else {
            printf("d");
            wait(NULL);
        }
    }
    return 0;
}
```

Note that `printf` statements are all atomic and flush to `stdout` immediately. The call `wait(NULL)` is equivalent to `waitpid(-1, NULL, 0)`.

1) List all possible output sequences from the program, in `alphabetical` order, assuming that every `fork()` call succeeds.

Note that not all blank spaces have to be filled with code. If a blank is unused, put a '-' in the blank instead. If there is a sequence of blank spaces, put your code starting from the first blank space available, and put '-' in any unused spaces left after.