1. Operating System Concept Chapter 4 Exercises: 4.8 4.9. 4.10 4.17 4.19 (50 points, 10 points each)

2. Compile and run the following program twice, with ‘vm’ and without ‘vm’ as the argument, respectively. Take the screenshots of the running results (20 points) and explain why the output is different (30 points).
   Hint: Understand the meaning of the CLONE_VM flag of the clone system call.

```
#define _GNU_SOURCE
#include <sched.h>
#include <sys/syscall.h>
#include <sys/wait.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <unistd.h>

static int child_func(void* arg) {
  char* buf = (char*)arg;
  printf("Child sees buf = \"%s\"\n", buf);
  strcpy(buf, "hello from child");
  return 0;
}

int main(int argc, char** argv) {
  // Allocate stack for child task.
  const int STACK_SIZE = 65536;
  char* stack = malloc(STACK_SIZE);
  if (!stack) {
    perror("malloc");
    exit(1);
  }

  // When called with the command-line argument "vm", set the CLONE_VM flag on.
  unsigned long flags = 0;
  if (argc > 1 && !strcmp(argv[1], "vm")) {
    flags |= CLONE_VM;
  }

  char buf[100];
  strcpy(buf, "hello from parent");
  if (clone(child_func, stack + STACK_SIZE, flags | SIGCHLD, buf) == -1) {
    perror("clone");
    exit(1);
  }

  int status;
  if (wait(&status) == -1) {
    perror("wait");
```

```
    exit(1);
  }

  printf("Child exited with status %d. buf = \"%s\"\n", status, buf);
  return 0;
}
```