Homework 1

CS211 - Fall 2024

(You should try to answer first and then compare your solution with the ChatGPT solution)

This homework is designed to help you become familiar with the C language, assembly language, and basic concepts of systems programming, such as the stack pointer, program counter, and interrupts.

Question 1: Install Linux and gcc/g++ on your computer if you are using Windows. For example, you can install Ubuntu by following this tutorial https://ubuntu.com/tutorials/install-ubuntu-desktop#1-overview

Question 2: We want to swap two numbers and two pointers. Check if the code below works and fix it if it does not

```
#include <stdio.h>
void swap_nums(int *x, int *y) {
     int tmp;
     tmp = *x;
     *x = *y;
     *y = tmp;
}
void swap_pointers(char *x, char *y) {
     char *tmp;
     tmp = x;
     x = y;
     y = tmp;
}
int main() {
    int a,b;
    a = 3; b=4;
    swap_nums(&a,&b);
    printf("a is %d\n", a);
    printf("b is %d\n", b);

    char *s1,*s2;
    s1 = "I should print second";
    s2 = "I should print first";
    swap_pointers(s1,s2);
    printf("s1 is %s\n", s1);
    printf("s2 is %s\n", s2);
    return 0;
}
```

Question 3: Find, explain, and eliminate any code that causes a segmentation fault

```
#include <stdio.h>
int main() {
     char **s;
     char foo[] = "Hello World";

     *s = foo;
     printf("s is %s\n",s);

    s[0] = foo;
    printf("s[0] is %s\n",s[0]);
    return(0);
}
```

Question 4: We want to retrieve the stack pointer (SP) and program counter (PC) to check if they are allocated as described in the lecture. In C, we cannot directly access this information, but we can use inline assembly to obtain it. Below is the code to do so.

```
    void* stack_pointer;
    void* program_counter;

    // Inline assembly to get the stack pointer (SP)
    __asm__("mov %%rsp, %0" : "=r"(stack_pointer));     // x86
    __asm__("mov %0, sp" : "=r"(stack_pointer));        // ARM

    // Inline assembly to get the program counter (RIP)
    __asm__("lea (%%rip), %0" : "=r"(program_counter));   // x86
    __asm__("mov %0, pc" : "=r"(program_counter));        // ARM
```

Your tasks are:

(a) Understand the code above

(b) Use the code above to write a program to print out SP and PC to see if SP is always larger than PC, as discussed in the lecture

Question 5: How can we determine the current location of the heap? Write a simple code to find the current location of the heap and the current stack pointer (SP), then calculate the size of the unused memory in KB or MB.

Question 6: The code below is to handle the interrupt caused by Ctrl + C.

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

// Function to handle the SIGINT signal (Ctrl+C)
void handle_sigint(int sig) {
    printf("We handle the interrupt %d here\n", sig);
    exit(0); // try replacing this with something else, e.g. printf("Hello!")
}

int main() {
    // Set up the SIGINT signal handler
    signal(SIGINT, handle_sigint);

    // Main loop
    while (1) {
        printf("Running... Press Ctrl+C to interrupt.\n");
        sleep(1); // Sleep for 1 second
    }

    return 0;
}
```

Your tasks are:

(a) Search for documentation on the functions used in the code and predict what the code does.

(b) Run the code and modify the bold line to validate your prediction