CSE 340 SPRING 2019

HOMEWORK 6

Due Wednesday, 4/24/2019 by 11:59 pm

- You should write your name on your submission.
- Remember that late submissions are not accepted for homework.
- **You should answer the questions in the order they are asked**
- **You should submit a single pdf file for the solution not multiple files.**

Problem 1. Consider the following code in C syntax:

```c
#include <stdio.h>

int temp = 0;

int sum(int i, int m, int n, int ai)
{
    for (i = m; i < n; i++) {
        temp = temp + ai;
    }
    return temp;
}

int main()
{
     int a[9] = {1,10,10,10,100,100,100,100,100};
     int b[4][4] = {0,1,2,3,4,5,6,7,8,9,10,11,12,13,14,15};

     int temp = 4;
     int m = 0;
     int n = 3;
     int i = 1;
     int j = 1;
     int result;

     result = sum(i, m , n, i);
     printf("%d %d\n", temp, result);
     for (int i = 0; i < 9; i++)
         printf("%d ", a[i]);
     printf("%d\n", a[i]);

     result = sum(i, m , n, a[i]);
     printf("%d %d\n", temp, result);
     for (int i = 0; i < 9; i++)
         printf("%d ", a[i]);
     printf("%d\n", a[i]);

     n = 2;
     result = sum(i, m , n, sum(j, m, n, b[i][j]));  // 1
     printf("%d %d\n", temp, result);                // 2
     for (int i = 0; i < 3; i++)                     // 3
         printf("%d ", a[i]);                        // 4
     printf("%d\n", a[i]);                           // 5
}
```

- What is the output of the program if functions are called *by value*?
- What is the output of the program if functions are called *by reference*? For this part do not execute lines //1 through //5
- What is the output of the program if functions are called *by name*?

Problem 2. Consider the following program written in Ada syntax with the execution stack shown on the right side. The line numbers are used to refer to the code and are not part of the code.

```
procedure env is
        x_env: integer;
        y_env: integer;

        procedure a is
                x_a: integer;
                procedure d is
                begin
                        b;
                end d;
        begin
                d;
        end a;

        procedure b is
                x_b: integer;
                procedure c is
                        x_c: integer;
                        procedure e is
                        x_e : integer;
                        begin
                          x_e = x_c + x_env ;
                          x_e = x_b ;
                          env;
                        end;
                begin
                        e;
                end c;
        begin
                c;
        end b;

begin
        a;
end env;
```

```
                 _____________
env   |_____________| 1
 a    |_____________| 2
 d    |_____________| 3
 b    |_____________| 4
 c    |_____________| 5
 e    |_____________| 6
env   |_____________| 7
 a    |_____________| 8
 d    |_____________| 9
 b    |_____________| 10
 c    |_____________| 11
 e    |_____________| 12
```

- Give the low-level code (in terms of mem[]) to setup the access link for activation record 5
- Give the low-level code (in terms of mem[]) to setup the access link for activation record 6
- Give the low-level code (in terms of mem[]) to setup the access link for activation record 7
- Give the low-level code (in terms of mem[]) to setup the access link for activation record 8
- Give the low-level code (in terms of mem[ ]) for `x_e = x_c + x_env;` in procedure e
- Give the low-level code (in terms of mem[ ]) for `x_e = x_b;` in procedure e

Note on Ada syntax. A procedure declaration has the following form

```
procedure p is

        // declaration of local variables and other
        // procedures nested within p
```

```
begin
        // body of p
end p;
```

procedure calls are written without parentheses. So, in the code above, the body of env has a call to procedure a in its body.