WEEKLY-EXERCISE - 10

ICS 365-51 **Metropolitan State University/MN**

Week 12 **Due 11:59pm, Sunday, Nov. 13th, 2022** **Fall 2022**

Name:_____Pong Lee_____ **Score: _____**

Please complete both Parts I and II and then upload the results to D2L under the dropbox for Weekly Exercise 10 before the deadline (total 20 points).

Part I: Based on the discussion in this week's lecture, please either **bold** or highlight your answers below, only one answer per question. (1 point each, total 10 points)

1. Based on the discussion in Chapter 13, which of the following might not be one of the standard methods provided by Java Thread class?

A) `fetch()`
B) `join()`
C) `run()`
D) `start()`

2. Based on the discussion in Chapter 13, which of the following statements is not true?

A) High-Performance Fortran provides statements for specifying how data is to be distributed over the memory units connected to multiple processors;

B) Physical concurrency refers to the situation when multiple processors are used to execute concurrent units;

C) Logical concurrency is defined as that concurrent united are executed on a single processor;

D) Concurrent execution can be at either instruction or statement level, but not at subprogram level.

3. Based on the discussion on Ada Message Passing in Chapter 13, which of the following statements is not true?

A) During execution of the accept clause, the sender is suspended;
B) Every accept clause has an associated queue to store waiting messages;
C) A task that has "accept" clauses, but no other code, is called an "actor" task;
D) The task executes to the top of the accept clause and waits for a message.

4. Based on the discussion in Chapter 13, which of the followings is an entry-point in the Ada task code presented on slide 47 or page 558 of the textbook?

A) `Next_In := (Next_In mod Bufsize) + 1;`
B) `Deposit(Item : in Integer)`
C) `Filled < Bufsize =>`
D) `Filled := Filled + 1;`

5. In which of following programming languages discussed in Chapter 13, "FORALL" statement is used to specify a list of statements that may be executed concurrently?

A) Ada
B) C++
C) High-Performance Fortran
D) Java

6. If `h(y) = y / 2` and `g(y) = y - 1`, what is the value of `h(g(y))` when `y = 5` ?

A) `-2`
B) `0`
C) `1`
D) `2`
E) None of above

7. Based on the discussion in Chapter 15, which of the followings is not one of the fundamentals of functional programming languages?

A) Variables are not necessary;
B) To mimic mathematical functions to the greatest extent possible;
C) Operations are done and the results are stored in variables for later use;
D) The evaluation of a function always produces the same result given the same parameters
message passing.

8. Based on the discussion on "Apply-to-All" function in Scheme, `(map (LAMBDA (x) (+ x (* x x))) '(1 2 3))` will yield _____.

A) (2 6 12)
B) (2 6 14)
C) (2 8 16)
D) (4 8 16)
E) Your answer _____

9. Based on the discussion in Chapter 15, which of the following statements is not true regarding lambda expressions?

A) Lambda expressions, like other function definitions, can have more than one parameter;
B) A lambda expression specifies the parameters and the mapping of a function;
C) Lambda expressions are applied to parameter(s) by placing the parameter(s) after the expressions;
D) A lambda expression can be defined as a function with a user-provided name.

10. Based on the discussion in Chapter 15, which of the following statements is <u>not</u> true to functional programming language "Scheme?"?

A) It is one of the functional programming languages;
B) It is a complete different from Lisp, a well-known functional programming language;
C) It was developed in 1970s;
D) It uses only static scoping.

Part II: Please study the lecture slides and handout covered this week before working on the following tasks: (Total 10 points)

2.1) Please provide the screenshot for the following calculations with LISP (clisp on our server), 1 point each for total 5 points):

a) `4 + 6 * 2`

b) `5 - 6 / 2 * (3 - 1)`

c) `(5 - 2) * 6 / 3 * (3 - 1)`

d) Please define a LAMBDA function for $a + 2b + c$ and then evaluate it with $a = 1, b = 2,$ and $c = 3$

e) Please define a LAMBDA function for $x^2 + 2xy + y^2$ and then evaluate it with $x = 2$ and $y = 4$

```
Type :h and hit Enter for context help.

[1]> (+ 4 ( * 6 2))
16
[2]> (- 5 (* ( / 6 2) (- 3 1)))
-1
[3]> (* (* (- 5 2)(/ 6 3)(- 3 1)))
12
[4]>
[4]> ((lambda (a b c) (+ a (+ c(* 2 b)))) 1 2 3 )
8
Break 1 [7]> ((lambda (x y) (+ (+ (expt x 2)(* (* 2 x)y)(expt y 2))))2 4)
36
```

2.2) Please provide the screenshot of the last C program (Semaphore with C Threads) provided in Handout A for Week 12. Please provide the list of the program (using "*cat*" command), the compiling command, the execution command, and the output in a single screenshot as shown in the handout. A better layout of the program is provided on the next page. (5 points):

```c
#include <pthread.h>
#include <semaphore.h>
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>

#define NITER 3
int ggg = 0;

sem_t sss;

void* Count(void* a)
{
        int i, tmp;
        for(i = 0; i < NITER; i++)
        {
                sem_wait (&sss);
                tmp = ggg;
                tmp = tmp + 1;
                sleep(2);
                ggg = tmp;
                printf("\n The current value of ggg is [%d] with the thread [%u].\n", ggg, (unsigned int)pthread_self());
                sem_post (&sss);
        }
}

int main(int argc, char * argv[])
{
        pthread_t tid1, tid2;

        sem_init(&sss, 0 ,1);

        if(pthread_create(&tid1, NULL, Count, NULL))
        {
                printf("\n ERROR creating thread 1");
                exit(1);
        }

        if(pthread_create(&tid2, NULL, Count, NULL));
        {
                printf("\n ERROR creating thread 2");
                exit(1);
        }

        if(pthread_join(tid1, NULL))
        {
                printf("\n ERROR joining thread");
                exit(1);
        }

        if(pthread_join(tid2, NULL))
        {
                printf("\n ERROR joining thread");
                exit(1);
        }

        if (ggg < 2 * NITER)
                printf("\n A wrong value for ggg = [%d]. It should be [%d]. \n\n", ggg, 2*NITER);
        else
                printf("\n Good! ggg= [%d]. \n\n", ggg);

        pthread_exit(NULL);
}
```

```
ics365fa2215@sp-cfsics:~/wk12$ pico semaphoreT3.c
ics365fa2215@sp-cfsics:~/wk12$ gcc -o semaphoreT3 semaphoreT3.c -l pthread
ics365fa2215@sp-cfsics:~/wk12$ ./semaphoreT3

 ERROR creating thread 2ics365fa2215@sp-cfsics:~/wk12$
```

- Example 3 from Handout A for Week 12:

```
[ics365fa2235@sp-cfsics:~/wk12$ cat semaphoreT3.c
#include <pthread.h>
#include <semaphore.h>
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>

#define NITER 3
// declare a global variable ggg
int ggg = 0;

// declare a semaphore variable sss
sem_t sss;

void * Count(void * a)
{
    int i, tmp;
    for(i = 0; i < NITER; i++)
    {
        sem_wait (&sss);
        tmp = ggg;      /* copy the global ggg locally */
        tmp = tmp+1;    /* increment the local copy */
        sleep(1);
        ggg = tmp;      /* store the local value into the global ggg */
        printf("\n The current value of ggg is [%d] with the thread [%u].\n", ggg, (unsigned int)pthread_self());
        sem_post (&sss);      /* signal sss */
    }
}

int main(int argc, char * argv[])
{
    pthread_t tid1, tid2;
//  initialize sss
    sem_init(&sss, 0, 1);

    if(pthread_create(&tid1, NULL, Count, NULL))
    {
      printf("\n ERROR creating thread 1");
      exit(1);
    }

    if(pthread_create(&tid2, NULL, Count, NULL))
    {
      printf("\n ERROR creating thread 2");
      exit(1);
    }

    if(pthread_join(tid1, NULL))        /* wait for the thread 1 to finish */
    {
      printf("\n ERROR joining thread");
      exit(1);
    }

    if(pthread_join(tid2, NULL))        /* wait for the thread 2 to finish */
    {
      printf("\n ERROR joining thread");
      exit(1);
    }

    if (ggg < 2 * NITER)
        printf("\n A wrong value for ggg = [%d]. It should be [%d].\n\n", ggg, 2*NITER);
    else
        printf("\n Good! ggg = [%d].\n\n", ggg);

    pthread_exit(NULL);
}
[ics365fa2235@sp-cfsics:~/wk12$
```