CS19001 Programming and Data Structures Lab

LAB TEST – 3 **June 30, 2021**

1. **Task Scheduling.** A set of *n* tasks, T_1 … T_n, will run on a single machine. The machine runs one task at a time and completes one task before starting the next. Each task T_k has a known execution time, R_k, which is the time for which it runs on the machine. If the output of task T_a is an input to task T_b, then T_b cannot be started before T_a has completed. Suppose the runtimes and the dependencies between the tasks are as shown in the following example:

   T_1, 10, T_2, T_3 *// The runtime of T_1 is 10, and it needs the outputs of T_2 and T_3*

   T_2, 5 *// The runtime of T_2 is 5. It has no input dependencies*

   T_3, 2

   T_4, 5, T_1, T_5

   T_5, 2, T_3 *// The runtime of T_5 is 2, and it needs the outputs of T_3*

   We can schedule these tasks on the machine in different ways such that they do not violate their input dependencies.

   For example, one valid schedule is [T_2, T_3, T_5, T_1, T_4]. Another valid schedule is [T_3, T_2, T_1, T_5, T_4].

   Write a program that does the following:

   (a) It reads the number of tasks, *n*, and then reads the runtimes and the input dependencies for each of the *n* tasks. Assume that no task has more than two input dependencies. Also assume that the input consists of *n+1* lines. The first line contains a single integer specifying the value of *n.* The kth line after the first line contains an integer specifying the runtime of task T_k followed by two integers for the input dependencies. If a task has less than two dependencies, then the user enters –1 in place of the remaining integer(s). For example, the input for our example will be:

   ```
   5
   10  2  3
   5  –1 –1
   2  –1 –1
   5   1  5
   2   3 –1
   ```

   The data should be read into a dynamically allocated array of *n* structures of the following type:

   ```
   struct task {
      int runtime;      // Runtime of the task
      int id1;          // Input dependency 1
      int id2;          // Input dependency 2
   }
   ```

   (b) The program should print the total time required to complete all tasks

   (c) The program should read an integer, *a*, and print the following:

   (i) The earliest time when task T_a can be scheduled. [Hint: *Identify the tasks that must precede T_a and sum their runtimes.*]

   (ii) The list of tasks that cannot be scheduled before T_a is completed.

[Answers for the given example when a=1:

Total time = 24

Earliest time when T_1 can start = 7 (runtime of T_2 + runtime of T_3)

List of tasks that cannot be scheduled before T_1 = { T_4 }]

[Submission Filename: ⟨Your roll number⟩LT3.c

If your roll number is 20CS30099, then the filename for this task will be 20CS30099LT3.c]