ENPM809X Homework #3
due April 06, 2023 4:00pm

1) (30 points) Suppose you have a hash table of length $m = 16$, implemented using open addressing. Show the result of inserting the set of keys {42, 45, 7, 61, 32, 4, 13, 27, 48} into the table using linear probing, quadratic probing, and double hashing, and calculate the number of collisions for each of the three methods. Use the following hash functions:

Linear probing: $h(k, i) = (k + i) \bmod m$

Quadratic probing: $h(k, i) = (k + i/2 + i^2/2) \bmod m$

Double hashing: $h(k, i) = (k + ih_2(k)) \bmod m$, where $h_2(k) = \big(k \bmod (m - 1)\big) + 1$

2) (20 points) In the Tree-Delete procedure, when there are two children we selected the successor of node *z* to replace *z*. Instead, we can choose the predecessor of node *z* to replace it. Modify the Tree-Delete pseudo-code so that node *y* is selected as the predecessor of *z* and the pointers are updated correctly as *y* replaces *z*.

3) (20 points) Consider red-black trees with exactly 9 nodes (excluding the “T.NIL”s). Draw a tree with the maximum number of red nodes, and another tree with the maximum number of black nodes. (You don’t need to worry about the key values – assume they are always in order) How many red and black nodes does each case have?

4) (from Problem 16-2) (30 points) Suppose we are given n tasks to be scheduled on a single processor. Each task has a known processing time to complete, given by $\{p_1, p_2, \dots, p_n\}$. Assume the processor is initially idle at time $t = 0$, and let t_i be the time *i*th task is completed. For example, if there are only two tasks with processing times $p_1 = 3$ and $p_2 = 5$, there are two possibilities for completion times:

- If the first task is scheduled first, then $t_1 = 3$ and $t_2 = 3 + 5 = 8$ (since it waited for the first one)
- Otherwise, $t_2 = 5$ and $t_1 = 5 + 3 = 8$

Our goal is to minimize the average completion time, given by

$$\frac{1}{n}\sum_{i=1}^{n} t_i$$

a) (15 points) Assuming tasks cannot preempt each other and all tasks are ready to be started at $t = 0$, give a greedy algorithm to minimize average completion time. Show that the algorithm achieves the optimal result.

b) (15 points) Now assume that tasks can preempt each other, i.e. a given task may start-stop-start-stop-… multiple times until it completes when the total time it is scheduled equals p_i. Also assume that tasks are <u>not</u> ready to be started at $t = 0$, but have arrival times given by $\{r_1, r_2, \ldots, r_n\}$, i.e. task *i* cannot start before $t = r_i$. Give a greedy algorithm to minimize average completion time for this case and show that it achieves the optimal result.