COP 3502 Section 4 Exam 1B – Recursion, Sums/Recurrences, Alg. Anl. (Thursday 10/12/2023)
Solution

1) (10 pts) Consider the problem of traveling on an n by n grid from an arbitrary position (r, c) to the bottom right corner (n-1, n-1) where there are two possible moves – moving down one row to (r+1, c) or moving to the right one column to (r, c+1). Let the grid be a 2D character array, where each character is either '.' or 'X', where '.' is a square you are allowed to travel to and 'X' is forbidden. Write a recursive function that counts the number of ways you can travel from (r, c) to (n-1, n-1). You may assume that grid[r][c] contains '.' and that grid[n-1][n-1] also contains '.'

```
#include <stdio.h>

int numways(char** grid, int n, int r, int c) {

    // Grading: 2 pts
    if (r == n-1 && c == n-1) return 1;

    // 1 pt for variable declarations, etc.
    int res = 0;

    // 2 pts for recursive call, 1 pt to avoid AOOB and check .
    if (r < n-1 && grid[r+1][c] == '.')
        res += numways(grid, n, r+1, c);

    // 2 pts for recursive call, 1 pt to avoid AOOB and check .
    if (c < n-1 && grid[r][c+1] == '.')
        res += numways(grid, n, r, c+1);

    // 1 pt
    return res;

}
```

2) (10 pts) Question 3 of Part A of this exam describes an algorithm to reverse a linked list. For this question you will analyze the run-time of that algorithm. Let T(n) be the run-time of the algorithm described in question 3 of Part A of this exam on a linked list of size n.

(a) (2 pts) Write down a recurrence relation, based on the description of the algorithm that T(n) satisfies.

T(n) = T(n – 1) + O(n), **Grading: 1 pt for each term on RHS**

(b) (2 pts) Briefly, justify the recurrence relation you wrote in part (a).

Step 1 is O(1) time, a single return. Step 2 is T(n-1) time because it's a recursive call on an list of size n-1. Step 3 loops through a list of size n-1, which takes O(n) time. Steps 4, 5 and 6 are all O(1) time. If we add up each step that isn't recursive, the dominating term is O(n). It follows that T(n) = T(n – 1) + O(n).

Grading: 1 pt for point out recursive call is on list of size n-1
1 pt for pointing out that the dominating extra work is a loop through a list of size n-1. (It's okay if they just say n also…) No need to mention the O(1) stuff.

(c) (6 pts) Using the iteration technique, determine a Big-Oh solution to the recurrence relation T(n), hence establishing the run-time of the algorithm described in question 3 of Part A of this exam to reverese a linked list of size n.

For simplicity, let's just solve the recurrence relation T(n) = T(n – 1) + cn, for a constant c. We can use T(1) = 1 as an initial condition.

$$T(n) = T(n-1) + cn$$
$$T(n) = T(n-2) + c(n-1) + cn$$
$$T(n) = T(n-3) + c(n-2) + c(n-1) + cn$$

Thus, after k steps we have:

$$T(n) = T(n-k) + c\sum_{i=k+1}^{n} i$$

Since we know T(1), plug in k = n - 1:

$$T(n) = T(1) + \sum_{i=2}^{n} ci = 1 + \frac{cn(n+1)}{2} - c = O(n^2)$$

Grading: 2 pts for iteration #2 and #3,
1 pt general guess
1 pt what to plug in for k.
2 pts to finish it up

3) (7 pts) Determine, as a function of n, the sum below**. <u>Please provide your answer in standard polynomial form.</u>**

$$\sum_{i=7}^{n+1}(2i+3)$$

$$\sum_{i=7}^{n+1}(2i+3)=\sum_{i=1}^{n+1}(2i+3)-\sum_{i=1}^{6}(2i+3)$$

$$=\frac{2(n+1)(n+2)}{2}+3(n+1)-(\frac{2(6)(7)}{2}+3(6))$$

$$=(n+1)(n+2)+3n+3-60$$

$$=n^2+3n+2+3n+3-60$$

$$=n^2+6n-55$$

Grading: 1 pt split sum, 1 pt each for each of the four formulas or just adding all terms for the last two sums, 2 pts to simplify all the way to the end.

Alternate solution: Recognize the sum is an arithmetic sequence with n – 5 terms, with first term 17 and last term 2(n+1)+3 = 2n + 5. It follows that the desired sum is:

$$\left(\frac{17+2n+5}{2}\right)\times(n-5)=\left(\frac{2n+22}{2}\right)(n-5)=(n+11)(n-5)=\boldsymbol{n^2+6n-55}$$

Grading: 1 pt # of terms, 1 pt first term, 1 pt last term, 3 pts for simplification

4) (7 pts) An algorithm with a run-time of O($n\sqrt{n}$) takes .8 seconds to run on an input of size n = 160,000. How long will the algorithm take when run on an input of size n = 1,000,000? **Please answer in seconds, rounded to the nearest tenth of a second.**

Let the run time of the algorithm be $T(n) = cn\sqrt{n}$, for some constant n. Using the given information, we have:

$$T(160000) = c(160000)\sqrt{160000} = .8\ sec$$

$$c = \frac{.8\ sec}{160000 \times 400} = \frac{.8\ sec}{400^3}$$

Plug in n = 1,000,000 = 1000^2:

$$T(1000^2) = \frac{.8\ sec}{400^3} \times 1000^2\sqrt{1000^2} = (.8\ \text{sec}) \times \left(\frac{1000}{400}\right)^3$$
$$= (.8\ \text{sec}) \times \left(\frac{5}{2}\right)^3 = \frac{4}{5} \times \frac{125}{8} = \frac{25}{2} = \mathbf{12.5\ sec}$$

Grading: 2 pts set up initial equation, 2 pts solve for c, 1 pt plug in new value, 2 pts arrive at final answer. (Only give full credit if answer is in the right form.)

5) (1 pt) What type of food is served at BurgerFi? **Burgers (give to all)**