Below are several exercises that we will be covering in the upcoming tutorial (Friday, Mar 25). We will release these exercises in advance of the tutorial so that you get a chance to attempt the exercises yourself before we discuss them in tutorial. The course staff running the tutorial will go through each problem (time permitting) and show their process for how they would complete these questions in an assignment/exam setting. Being familiar with the questions before attending tutorial will help to ensure that you get the most out of our explanations.

Note that tutorials are not mandatory in CS135. We never cover any new material that you won’t have seen in lectures, and instead choose to highlight techniques and concepts from the most recent lectures to give you some extra practice. If you are very comfortable with the material that we covered and could easily complete the below questions, then you aren’t obligated to attend tutorials. Be very cautious with this though since sometimes questions can seem easy until you actually sit down and attempt them.

**Question 1: Archery Scores**

You’ve been asked to automate the scoring process for an upcoming archery competition.

In the competition, each archer will shoot multiple rounds of arrows at a target. Each round consists of multiple shots. Each time the archer shoots, their arrow may hit the target or miss. If the arrow hits, it may hit dead center (bullseye) or hit somewhere else on the target (hit).

Write a function \texttt{(archery-scores rounds)} which consumes a list of rounds for one archer in the competition. Each round is a list of symbols, where each symbol represents one of three possible outcomes of one arrow shot by the archer:

- ‘bullseye, which receives a score of 2
- ‘hit, which receives a score of 1
- ‘miss, which receives a score of 0

The function should produce a single value: the sum of all points received by the given archer.

\textbf{Important:} You must only use cond, lambda, and either foldr or foldl in your solution.

\textit{Example:}

\texttt{(archery-score (list (list 'hit 'miss 'bullseye) (list 'hit 'hit) (list 'miss))) → 5}

**Question 2: Cube and Square Differences**
Using build-list, write a function \((\text{cubes-minus-squares } n)\) which consumes a natural number \(n\) and produces the difference between the cube and the square of each value from 0 to \(n-1\).

**Examples:**

\((\text{cubes-minus-squares } 0) \rightarrow \text{empty}\)

\((\text{cubes-minus-squares } 5) \rightarrow \text{(list 0 0 4 18 48)}\)

**Question 3: Stop, Thief!**

Quick! A thief is escaping down a maze of streets and you need to catch them!

You know the \(x,y\) coordinates of the position where the thief started and the direction they were facing, but lost sight of the thief after that. Thankfully, people are watching and reporting the thief’s movements, so you are given a list of integers representing the path taken by the thief. Each integer represents one sighting of the thief \((S)\), which may be one of:

- \(S = -1:\) the thief was seen turning \textit{left} onto a new street!
- \(S = 0:\) the thief was seen turning \textit{right} onto a new street!
- \(S \geq 1:\) the thief was seen running \(S\) steps \textit{straight} down the street!

Write a function \((\text{catch-thief path } x\ y\ \text{dir})\) which consumes:

1. A list of integers representing the thief’s reported movements.
2. The \(x\) and \(y\) coordinates representing the thief’s starting position.
3. The direction the thief starts facing, which may only be one of ‘north’, ‘south’, ‘east’, or ‘west’.

The function produces a list of exactly two integers: the \(x\) and \(y\) coordinates of the thief after executing the full path.

**Important:** Your solution must use generative recursion.

**Examples:**

\((\text{catch-thief (list } 5\ 10\ 15\ 0\ 6\ 6\ -1\ 2\ 2\ 4\ 1\ 1)\ 0\ 0\ \text{‘north}) \rightarrow \text{(list 12 40)}\)

- Explanation: The thief started at \((0,0)\), ran 30 steps north, turned right, ran 12 steps east, turned left, and ran 10 more steps north before stopping, so her final position is \(x=12\) and \(y=40\).

\((\text{catch-thief (list } 10\ 0\ 10\ 0\ 10\ 0\ 10)\ 0\ 0\ \text{‘north}) \rightarrow \text{(list 0 0)}\)

- Explanation: The thief started at \((0, 0)\) and ran 10 steps north, east, south, and west to return to his starting position.