Due Friday 7 April, *before* 1:00pm

General instructions for Problem Sets

Please read the following instructions carefully before starting the problem set. They contain important information about general problem set expectations, problem set submission instructions, and reminders of course policies.

- Your problem sets are graded on both correctness and clarity of communication. Solutions that are technically correct but poorly written will not receive full marks. Please read over your solutions carefully before submitting them.
- Solutions must be typeset and submitted as a PDF with the correct filename. **Handwritten submissions will receive a grade of ZERO.**

  The required filename for this problem set is **problem_set4.pdf**.
- Each problem set may be completed in groups of up to three—**except for Problem Set 0**. If you are working in a group for this problem set, please consult https://github.com/MarkUsProject/Markus/wiki/Student-Guide for a brief explanation of how to create a group on MarkUs.
- Problem sets must be submitted online through MarkUs. If you haven't used MarkUs before, give yourself plenty of time to figure it out, and ask for help if you need it! If you are working with one or more partner(s), you must form a group on MarkUs, and make one submission per group.
- Your submitted file(s) should not be larger than 19MB. You might exceed this limit if you use a word processor like Microsoft Word to create a PDF; in that case, you should look into PDF compression tools to make your PDF smaller, but please make sure that your PDF is still legible before submitting!
- Submissions must be made *before* the due date on MarkUs. You may use *grace credits* to extend the deadline; please see the course syllabus for details on using grace credits.
- MarkUs may be slow when many students try to submit right before a deadline. **Aim to submit your work at least one hour before the deadline. It is your responsibility to meet the deadline.** You can submit your work more than once (and you are encouraged to do so); the most recent version submitted within the deadline (or within the late submission period) is the version that will be marked.
- The work you submit must be that of your group; you may not use or copy from the work of other groups, or external sources like websites or textbooks. Please see the section on Academic Integrity in the course syllabus for further details.

Additional instructions

- All final Big-O, Omega, and Theta expressions should be fully simplified according to three rules: no constant factors (e.g., use $\mathcal{O}(n)$, not $\mathcal{O}(3n)$), no slower-growing terms (e.g., use $\mathcal{O}(n^2)$, not $\mathcal{O}(n^2 + n)$), and no floor or ceiling functions (e.g., use $\mathcal{O}(\log n)$, not $\mathcal{O}(\lceil \log n \rceil)$).
- For algorithm analysis questions, you can jump immediately from a closed-form step count expression to an asymptotic bound without proof (e.g., write "the number of steps is $3n+\log n$, which is $\Theta(n)$"). This applies to upper and lower bounds (Big-O and Omega) as well.
- However, you must simplify all summations *before* jumping to an asymptotic bound.

1. [7 marks] Nested loops.

Consider the following algorithm. (It doesn't **return** anything useful, but it sure spends a bunch of time before returning!)

```
def loop_de_loop_de_loop(n: int) -> None:
    """ Precondition: n > 0 """
    i = 0
    s = 1
    while i < n:                # Loop 1
        i = i + s
        s = s + 2
        j = 0
        while j < i:            # Loop 2
            k = n
            while k > 1:        # Loop 3
                k = k // 2
            j = j + 3
```

(a) **[3 marks]** Let $\mathtt{s}_t$ be the value of variable `s` and $\mathtt{i}_t$ be the value of variable `i` immediately *after* t iterations of Loop 1 have occurred. Determine a general formula for each of $\mathtt{s}_t$ and $\mathtt{i}_t$ and then find the exact number of iterations of Loop 1 for a given value of `n`.

(b) **[4 marks]** Give a Theta bound on the running time function $RT(n)$ for this algorithm. Show your work. (That is, explain how you obtained your answer and show your calculations).

2. **[8 marks] Worst-case analysis.**
Consider the following algorithm.

```
def algo(lst: list[int]) -> None:
    """ Precondition: len(lst) > 0 """
    n = len(lst)
    i = 0
    j = 0
    while i < n:
        if lst[i] % 3 == 0:
            j = j + i
            while j > 0:
                j = j // 2
            i = i + 2
        else:
            j = j + n
            i = i + 1
```

(a) **[4 marks]** Find, with proof, an **upper bound** on the **worst-case** running time of this algorithm. Show your work. For full marks, your upper bound must match the lower bound determined in the next part.

(b) **[4 marks]** Find, with proof, a **lower bound** on the **worst-case** running time of this algorithm. Show your work. For full marks, your lower bound must match the upper bound determined in the previous part.

3. [10 marks] Average-case analysis.

Consider the following algorithm.

```
def has_even(numbers: list[int]) -> bool:
    """Return whether numbers contains an even number.
    Precondition: len(lst) > 0
    """
    for number in numbers:
        if number % 2 == 0:
            return True
    return False
```

(a) **[1 mark]** Write a set of allowable inputs $\mathcal{I}_{\texttt{has\_even},n}$ for which $Avg_{\texttt{has\_even}}(n)$ is $\Theta(1)$. State briefly the reasoning used to arrive at your answer.

(b) **[1 mark]** Write a set of allowable inputs $\mathcal{I}_{\texttt{has\_even},n}$ for which $Avg_{\texttt{has\_even}}(n)$ is $\Theta(n)$. State briefly the reasoning used to arrive at your answer.

(c) **[2 marks]** Define

$$S_n = \{\text{input lists } \texttt{numbers} \text{ to } \texttt{has\_even} \mid \forall \texttt{i} \in \texttt{range}(n), 1 \leq \texttt{numbers[i]} \leq n\}.$$

Note that an `int` value is allowed to be repeated in an element of S_n. For example,

$$S_2 = \{[1, 1], [1, 2], [2, 1], [2, 2]\}.$$

Consider the set of allowable inputs $\mathcal{I}_{\texttt{has\_even},n} = S_n$. Give an expression for $|\mathcal{I}_{\texttt{has\_even},n}|$. State briefly the reasoning used to arrive at your answer.

(d) **[6 marks]** Calculate an *exact* expression for $Avg_{\texttt{has\_even}}(n)$ for the set of allowable inputs $\mathcal{I}_n = S_n$, where S_n is as defined in the previous part. Show your work.

Hint: You may use without proof any helpful correct formula that simplifies an expression containing a summation.