Assignment 1 (10% of total marks)

Due date: Sunday, 26 January 2025 by 9:00 pm (21:00 hours) Singapore time.

Scope

The tasks of this assignment consist of exercises in the areas of **Data Structures and Algorithms**, focusing on algorithm complexity analysis and recursion algorithms. The assignment covers the topics discussed in **Topics 1, 2, and 3**.

The assignment consists of:

- **Part A**: Theory questions to assess your understanding of the subject matter and support the programming question's objective.
- **Part B**: Programming questions requiring the implementation of concepts related to algorithms and their complexity.

Marks for each question are indicated next to the respective question. The total mark for Assignment 1 is **100 marks**.

Marks Distribution

- **Part A**: 70%
- **Part B**: 30%
- **Total Marks**: 100
- **Weightage**: 10% of the total subject mark

Assessment Criteria

Marks will be awarded for:

- **Correctness**: Solutions must meet the requirements of the question.
- **Comprehensiveness**: Solutions should demonstrate a complete understanding of the topic.
- **Appropriate Application**: Answers should reflect proper application of the materials covered in this subject.

Deliverables

Part A: Theory Questions (Questions 1 to 3)

- Type your answers using **MS Word** and save the completed document as a **PDF file**.
- Alternatively, you may write your answers by hand onto a piece of paper or using tablet. Ensure your handwriting is **neat**, then scan or save your handwritten solutions into a **PDF file**.

Part B: Programming Question (Question 4)

- Implement your solution following the standard requirements provided for the question. You may use **C, C++, Java, or Python**.
- Execute your program and take a **screenshot of the output**.
- Save your source code in **pure ASCII text format** with the appropriate file extension (e.g., **.cpp** for C++, **.java** for Java, or **.py** for Python).
- Ensure your program is appropriately documented with comments.
- Provide clear **compilation instructions** in a readme.txt file. Include batch or makefiles if applicable for C or C++ programs.
    - *Note*: If no compilation instructions are provided, your lecturer will attempt compilation based on their environment. Any failure due to incompatibility will result in a "failure to compile" outcome.

Submission Instructions

- Combine the following into a single **ZIP file** named **SolutionA1.zip**:
    - Your solutions to the theory questions (in **PDF format**).
    - Your source code for the programming question (in **ASCII text format**).
    - The screenshot of your program's output.
- Do not include subdirectories in your submission.

Part A: Theory Questions (70 Marks)

Question 1 – Algorithm Complexity Analysis (15 Marks)

a) Arrange the following functions in increasing order; that is, $f(n)$ should come before $g(n)$ in your list if and only if $f(n)$ is $O(g(n))$. Note: $n \geq 0$. **Justify all your answers**.

- $2^{\log_2 \log_2 n}$,
- 2^{3n},
- $(n-3)!$,
- 10^{10},
- $\sqrt{16n^2}$,
- $\log_2 \log_2 (n+10)^n$,
- $27^{\log_3 n}$,
- $\sum_{k=1}^{n} k$,
- $\left(\sqrt{2}\right)^{\log_2 \log_2 n}$,
- $\log_2 (\log_2 n)^3$

(5.0 marks)

b) Determine the big O running time of the method myMethod() by counting the approximate number of operations it performs. **Show all details of your answer and justify**.

```
Static int doIt(int n) {
    $for\ i \leftarrow 1\ to\ 10\ do$
        $for\ k \leftarrow 1\ to\ n\ do$
            $j \leftarrow 1; m \leftarrow n$
            $while\ j < m\ do$
                $m \leftarrow (m+j)/2$
            $end\ while$
        $end\ for$
    $end\ for$
}
```

```
static int myMethod(int n) {
    $i \leftarrow 1$
    $while\ (i < n)\ \{$
        $doIt(n)$
        $i \leftarrow i \times 2$
    $\}$
    $return\ 1;$
}
```

(5.0 marks)

c) Is it true that $\frac{n^2}{2} + n \lg n \in O(5n(4 + 2^{\lg n}))$? Justify your anwer.

(5.0 marks)

Question 2 – Recursive Algorithm (25 Marks)

Your lecturer is a funny guy; given an unsorted list of integer numbers of n elements, and to find the sum of all the numbers in the list, he will use the following algorithms.

```
function Sum(A, left, right)
        if left > right:
                return 0
        else if left = right:
                return A[left]
        mid = floor((left+right)/2)
        lsum = Sum(A, left, mid)
        rsum = Sum(A, mid+1, right)
        return lsum + rsum + A[mid]
```

```
function myMethod(A, N)
        B = new Array of length 1
        B[0] = Sum(A, 0, N-1)
        return B
```

a) Analyse the asymptotic run-time complexity of the method myMethods. Give the worst-case and the best-case running time in terms of Θ notation. Justify your answer. **(15.0 Marks)**

b) Explain briefly, in words, why the best-case inputs and the worst-case inputs are the same for the myMethods. **(10.0 Marks)**

Question 3 – Master Theorem (30 Marks)

Solve the following recurrence equations. For all the recurrence equations listed below, $T(1) = 1$. You may use Master Theorem or expansion and substitution techniques.

(i) $T(n) = 4T\left(\frac{n}{4}\right) + \frac{n}{\log_2 n}$, $and\ T(1) = 1$ **(5.0 mark)**

(ii) $T(n) = 2T\left(\frac{n}{4}\right) + \sqrt{n},\ \ and\ T(1) = 1$ **(5.0 mark)**

(iii) $T(n) = 2T\left(\frac{n}{2}\right) + n\ lg^2 n,\ \ and\ T(1) = 1.$ **(5.0 mark)**

(iv) $T(n) = 2T\left(\frac{n}{2}\right) + n\ lg\ n\ ,\ \ and\ T(1) = 1.$ **(5.0 mark)**

(v) $T(n) = 2T\left(\frac{n}{2}\right) + n^3,\ \ and\ T = 1.$ **(5.0 mark)**

(vi) $T(n) = T(n-1) + 3n + 3, \text{and } T = 1.$ **(5.0 mark)**

Part B: Programming Question (30 Marks)

Question 4 (30 Marks)

a) Write a recursive function convertStringToInteger(<String parameter>) that returns the integer value represented by the given string. The function must handle the following cases:

1. **Valid Input:**
   - The string can contain numeric digits ('0' through '9'), optionally preceded by a single minus sign ('-') to indicate a negative number.
   - Example valid inputs: "123", "-456", "0".
2. **Invalid Input:**
   - If the string contains any characters other than digits and an optional leading minus sign, it is invalid, and the function must throw a string exception with an appropriate message.
   - Example invalid inputs: "12a3", "-12-34", "++123".
3. **Empty String:**
   - If the input string is empty, the function should return 0.
4. **Range Guarantee:**
   - You may assume that the input string represents a value within the range of a 32-bit signed integer (-2147483648 to 2147483647). You do not need to handle overflow or underflow explicitly.

Constraints:

- The function must use recursion to process the input string.
- Avoid using built-in functions for string-to-integer conversion.

Marking guide:

- Programs submitted must work! A program which fails to compile or run will receive a mark of zero.
- A program which produces the correct output, no matter how inefficient the code, will receive a minimum of 50%, that is, 25 marks. Additional marks beyond this will be awarded for the appropriateness, that is, efficiency of this program as well as the algorithms and data structures

you use.

- Programs which lack clarity, both in code and comments, will lose marks.

(20.0 Marks)

b) Determine the algorithm's time complexity of your algorithm using Big-Theta (Θ) notation. **(10.0 Marks)**

Notes on Submission

1. This assignment is due on **Sunday, 26 January 2025 by 9:00 pm (21:00 hours) Singapore time**.
2. Submission must be made via **Moodle**. Ensure all files are combined into a single **ZIP file** without subdirectories.
3. Clearly label all components of your submission.
4. Late submissions will incur a **5% deduction per day**, including weekends.
5. Submissions more than **seven** days late will not be marked unless an extension is granted.
6. Extensions must be applied for via **SOLS** before the assignment deadline.
7. Plagiarism is treated seriously. Instances of plagiarism may result in a zero mark for the assignment.

Submit the file **SolutionA1.zip** through Moodle in the following way:

1) Access Moodle at **http://moodle.uowplatform.edu.au/**
2) To login use a Login link located in the right upper corner the Web page or in the middle of the bottom of the Web page
3) When successfully logged in, select a site **CSCI203 (SP125) Algorithms and Data Structures**
4) Scroll down to a section **Submissions of Assignments**
5) Click at **Submit** your Assignment 1 here link.
6) Click at a button **Add Submission**
7) Move the files created into an area provided in Moodle. You can drag and drop files here to add them. You can also use a link *Add…*
8) Click at a button **Save** changes,
9) Click at check box to confirm authorship of a submission,
10) When you are satisfied, remember to click at a button **Submit assignment**.

A policy regarding late submissions is included in the subject outline. Only one submission per student is accepted.

Assignment 1 is an individual assignment, and it is expected that all its tasks will be solved individually without any cooperation with the other students. Plagiarism is treated seriously. Students involved will likely receive zero. If you have any doubts, questions, etc. please consult your lecturer or tutor during lab classes or over e-mail.

End of specification

If you have any questions or require clarification, please contact your lecturer as early as possible. Good luck!