VIETNAM NATIONAL UNIVERSITY - HO CHI MINH CITY
HO CHI MINH CITY UNIVERSITY OF TECHNOLOGY
FACULTY OF COMPUTER SCIENCE AND ENGINEERING

PROGRAMMING FUNDAMENTALS - CO1027

Assignment 1

ONE PIECE
Arc Water 7 - Enies Lobby (Part 1)

Version 1.0

Ho Chi Minh City, 02/2026

ASSIGNMENT SPECIFICATION

Version 1.2

1 Learning Outcomes

After completing this assignment, students will review and master the following concepts:

- Conditional structures
- Loop structures
- One-dimensional and two-dimensional arrays
- String processing
- Functions and function calls
- File input/output operations

2 Introduction

One Piece is a renowned series that revolves around the journey of Monkey D. Luffy and his companions as they pursue the legendary treasure known as One Piece, with the ultimate goal of becoming the Pirate King. Throughout the story, the work not only explores adventures and conflicts on the high seas but also emphasizes core values such as teamwork, loyalty, personal ambition, and the choice between justice and freedom. Each stage of the journey (arc) presents a new set of challenges, where the characters must make decisions, accept the consequences, and grow through each turning point.

In the Water 7 – Enies Lobby arc, upon arriving at Water 7—a place famous for its shipbuilding industry—the main characters face a series of successive upheavals. The ship Going Merry is deemed severely damaged and beyond repair; internal disagreements begin to surface; and the conspiracies involving the CP9 organization and Robin are gradually revealed. These events lead the group to Enies Lobby, where they are forced to overcome numerous dangerous trials in order to protect their comrades and continue their journey ahead.

Inspired by this narrative thread, the Major Assignment (BTL) is designed to simulate these events in a logical and strategic manner, thereby enabling students to apply and practice fundamental C++ programming techniques through a cohesive scenario with progressively increasing difficulty. The assignment does not retain the entire original storyline; instead, it uses key events as reference points to construct programming problems that suit the requirements of the task.

In BTL 1, the assignment focuses on the early phase of this journey, when events unfold in rapid succession and lead to fractures within the group, laying the groundwork for subsequent challenges and crucial decisions.

3 Input Data

3.1 Input Data Description

The input data of the program is contained in a file, whose name is stored in the variable **opw_input**. Each line follows the general format:

```
<NAME>␣<INDICES>
```

The file structure is as follows:

```
<NAME_1>␣<HP_1>␣<SKILL_1>
<NAME_2>␣<HP_2>␣<SKILL_2>
...
<NAME_7>␣<HP_7>␣<SKILL_7>
GOING_MERRY␣<SHIPHP>␣<REPAIRCOST>
```

Where:

- `NAME_i`: The name of the i-th character (a string without spaces, consisting of letters and the underscore _). Possible characters include: LUFFY, ZORO, SANJI, NAMI, CHOPPER, USOPP, ROBIN
- `HP_i`: The endurance value of the i-th character (integer).
- `SKILL_i`: The overall skill value of the i-th character (integer).
- `GOING_MERRY`: The line describing the ship (fixed keyword).
- `SHIPHP`: The integrity level of the ship (integer).
- `REPAIRCOST`: The cost to repair the ship (integer).

Note: The order of the lines is not fixed.

Data Constraints

- For each character:

  - $0 \leq$ `HP_i` ≤ 1000
  - $0 \leq$ `SKILL_i` ≤ 100
- For the ship:
  - $0 \leq$ `SHIPHP` ≤ 1000
  - $0 \leq$ `REPAIRCOST` ≤ 3000

If a read value is less than 0, it is set to 0; if it exceeds the limit, it is set to the corresponding upper bound. Any calculation requiring rounding to an integer must be rounded up to the nearest greater integer.

Illustrative Example:

```
LUFFY␣120␣95
ZORO␣110␣90
SANJI␣105␣88
NAMI␣80␣60
CHOPPER␣90␣55
USOPP␣70␣50
ROBIN␣85␣65
GOING_MERRY␣320␣1500
```

3.2 Task 0: Reading Input Data

3.2.1 Function Description

- **Function name:** `readInput`
- **Input parameters:**
  - `filename`: The name of the file containing the input data.
  - `character`: A two-dimensional array storing character names in the order of appearance.
  - `hp`: An array storing the endurance values corresponding to each character.
  - `skill`: An array storing the skill values corresponding to each character.
  - `shipHP`: The integrity value of the ship.
  - `repairCost`: The repair cost of the ship.
- **Return value:**
  - `true`: If the file is opened successfully and the data is read correctly.

  - `false`: If the file cannot be opened or the data is invalid.

- **Functionality:** Students are required to implement the `readInput` function to read data from the input file and store the information into the corresponding arrays.
  - Read each line of the input file sequentially.
  - Each line has the format `<NAME> <INDEX_1> <INDEX_2>`.
  - If `<NAME>` is a character name:
    * Store the name in the `character` array in order of appearance (no duplicates).
    * Store the corresponding indices in the `hp` and `skill` arrays.
    * If the character name has appeared before, update its indices with the latest values.
  - If `<NAME>` is `GOING_MERRY`:
    * Store the values in `shipHP` and `repairCost`.

  After reading is complete, the data in the arrays and variables is **guaranteed to be valid and compliant with the problem specifications**, ready for subsequent tasks.

3.2.2 Illustrative Example

Assume the input file contains the following content:

```
ZORO 110 90
GOING_MERRY 320 1500
LUFFY 120 95
USOPP 70 50
CHOPPER 90 55
NAMI 80 60
SANJI 105 88
ROBIN 85 65
```

After reading and processing the file, the arrays/variables have the following values:

- `character`: { ZORO, LUFFY, USOPP, CHOPPER, NAMI, SANJI, ROBIN }
- `hp`: {110, 120, 70, 90, 80, 105, 85}
- `skill`: {90, 95, 50, 55, 60, 88, 65}
- `shipHP`: 320
- `repairCost`: 1500

4 Tasks

4.1 Task 1: Evaluating the Condition of the Going Merry Ship

Water 7 is known for its renowned shipbuilding and repair facilities. The Straw Hat crew arrives at this location to assess the condition of the Going Merry ship and estimate the repair cost. The purpose of the function in this task is to determine whether the current integrity of the ship requires an adjustment to the initially estimated repair cost.

4.1.1 Function Description

- **Function name**: `damageEvaluation`
- **Input parameters**:
  - `shipHP`: The integrity value of the ship.
  - `repairCost`: The ship repair cost.
- **Return value**: The value of `repairCost`
- **Functionality**:
  We define the concept of a *perfect number*: a perfect number is a positive integer whose sum of its positive divisors (excluding itself) is equal to the number itself. For example, $6 = 1 + 2 + 3$ is a perfect number. The function evaluates the ship condition based on the following two conditions:
  1. `shipHP` is less than 455.
  2. The sum of the digits of `shipHP` is a **perfect number**.

  If both conditions are satisfied, the ship is considered to be in a severely damaged state and requires urgent repair; in this case, the `repairCost` is increased by 50%. In all other cases, the repair cost remains unchanged.

4.1.2 Illustrative Example

Example 4.1

Given the input data:

```
shipHP␣=␣411
repairCost␣=␣253
```

Evaluate the conditions:

1. $shipHP = 411 < 455$
2. $4 + 1 + 1 = 6$. Since 6 is a perfect number

Both conditions are satisfied; therefore:
$repairCost = 253 + 253 \times 0.5 = 379.5 \rightarrow 380$

4.2 Task 2: Conflict Between Luffy and Usopp

After the evaluation, the shipwrights conclude that the condition of the Going Merry is beyond repair. The entire crew is deeply saddened. Luffy decides to abandon the ship and continue the journey with a new vessel, while Usopp considers this decision a betrayal and empathizes with the Going Merry, as both are regarded as "worthless." The divergence in viewpoints between Luffy and Usopp begins to emerge and intensifies over time. This task simulates the accumulation of internal conflict by identifying occurring events and updating the corresponding conflict index.

4.2.1 Function Description

- **Function name**: `conflictSimulation`
- **Input parameters**:
  - `character[][MAX_NAME]`: An array containing the names of the 7 characters.
  - `hp[]`: An array storing the endurance values corresponding to each character.
  - `skill[]`: An array storing the skill values corresponding to each character.
  - `shipHP`: The integrity value of the ship.
  - `repairCost`: The ship repair cost.
  - `maxSteps`: The maximum number of simulation steps.
- **Return value**: The value of the conflict index after the simulation process ends.

- **Functionality**:

  The function simulates the escalation of conflict over discrete time steps. At each step, an event is determined based on the current pressure level by computing the conflict index `conflictIndex`. This value is an integer calculated using the following formula:

$$conflictIndex = skill\_Luffy - skill\_Usopp + \frac{repairCost}{100} + \frac{500 - shipHP}{50}$$

  The event that occurs is identified by the event ID:

$$id = conflictIndex \bmod 6$$

  The possible events are described in Table 1.

Table 1: Events That Increase the Conflict Index

Event ID	**Event Description**	**Increase in conflictIndex**
0	All events occur simultaneously.	Increase by 255
1	The crew receives unfavorable information regarding the condition of the Going Merry.	Increase by 20
2	Luffy decides to abandon the Going Merry in order to continue the journey.	Increase by 50
3	Usopp opposes Luffy's decision and considers it an act of betrayal.	Increase by 70
4	The disparity in abilities and roles between Luffy and Usopp becomes increasingly apparent.	Increase by 90
5	Internal tension reaches a high level, with conflict accumulating through successive events.	Increase by 100

After an event is applied, the process continues until the `conflictIndex` reaches at least `255` or a total of 10 events have been processed.

4.2.2 Example

Example 4.2

Assume:

$$skill_{\texttt{Luffy}} = 95, \quad skill_{\texttt{Usopp}} = 50, \quad repairCost = 1500, \quad shipHP = 350, \quad maxSteps = 10$$

At the first simulation step, we have:

$$conflictIndex = 95 - 50 + \frac{1500}{100} + \frac{500 - 350}{50} = 63$$

Thus:

$$id = 63 \bmod 6 = 3$$

According to Table 1, the event with ID 3 occurs and the conflict index increases by 70. Therefore, after applying the effect of this event:

$$conflictIndex_{\text{new}} = 63 + 70 = 133$$

The process continues for subsequent steps until the `conflictIndex` reaches at least 255 or a total of `10` events have occurred.

This implies that the conflict between Usopp and Luffy inevitably reaches its peak, leading to a confrontation (described in the next task).

4.3 Task 3: The Duel Between Luffy and Usopp

When the conflict index between Luffy and Usopp reaches a high threshold, Usopp decides to leave the crew and challenges Luffy to a duel for the ownership of the Going Merry. This confrontation is not intended to determine superiority in strength, but rather represents a clash of beliefs, pride, and emotional attachment to the ship. Although forced to accept the duel, Luffy deliberately limits the involvement of other crew members in order to prevent further damage to the cohesion of the group. This task simulates the process of selecting appropriate members to act as stabilizing supporters, thereby minimizing the negative impact of the confrontation on the crew as a whole.

4.3.1 Function Description

- **Function name**: `resolveDuel`
- **Input parameters**:
  - `character[][MAX_NAME]`: An array containing the names of the 7 characters.
  - `hp[]`: An array storing the endurance values corresponding to each character.
  - `skill[]`: An array storing the skill values corresponding to each character.
  - `conflictIndex`: The conflict index
  - `repairCost`: The repair cost of the Going Merry.
  - `duel`: Array of selected characters
- **Return value**: Assign the set of characters selected to provide indirect support to Luffy to the `duel` paramter.
- **Functionality**:
  In this task, the characters are divided into three groups:
  - **Luffy**: The main participant in the duel.
  - **Usopp**: The opponent facing Luffy.
  - **Members**: The remaining crew members (excluding Luffy and Usopp).

  For each remaining member i, determine:
  - Support level: $support_i = skill_i$
  - Intervention cost: $cost_i = (hp_i \bmod 10) + 1$

  Usopp's resistance in the duel is defined as:

$$U = skill_{\texttt{USOPP}} + \frac{conflictIndex}{20} + \frac{repairCost}{500}$$

  Luffy wins the duel if there exists a subset of members such that:

$$skill_{\texttt{LUFFY}} + \sum support_i \geq U$$

  Among all subsets that satisfy the condition above, the function must select the subset with the **minimum total intervention cost**, reflecting that combat is undertaken only as a last resort and with minimal emotional damage to the crew. If multiple subsets satisfy this criterion, priority is given to the subset with fewer members.

4.3.2 Illustrative Example

Example 4.3

Assume:

$$skill_{\texttt{LUFFY}} = 95, \quad skill_{\texttt{USOPP}} = 50, \quad conflictIndex = 120, \quad repairCost = 1500$$

We have:

$$U = 50 + \frac{120}{20} + \frac{1500}{500} = 59$$

Suppose two members are selected with support levels $support_1 = 20$ and $support_2 = 10$. Then:

$$skill_{\texttt{LUFFY}} + support_1 + support_2 = 95 + 30 = 125 \geq 59$$

Therefore, this subset guarantees Luffy's victory. Among all valid subsets, the one with the minimum total intervention cost is selected as the function result.

After Usopp loses the duel and decides to leave the crew, Luffy returns the Going Merry to him as an apology and as recognition of Usopp's emotional bond with the ship. Shortly thereafter, however, the Going Merry "dies," leaving the entire crew—especially Usopp—in deep regret.

4.4 Task 4: The Departure of Nico Robin

Before the internal rift has time to heal, Nico Robin suddenly leaves the crew in silence. Only afterward does the Straw Hat crew gradually realize that all the incidents at Water 7 trace back to a single name: CP9—the secret intelligence agency of the World Government.

Before leaving, Robin secretly leaves behind a message. By decoding this message, the crew can uncover that the incident is connected to CP9 and the World Government. To avoid detection, Robin embeds the correct message among numerous meaningless decoy messages. A message is considered valid if it contains the substring "CP9" or "ENIESLOBBY".

4.4.1 Function Description

- **Function name**: `decodeCP9Message`
- **Input parameters**:
  - `character[][MAX_NAME]`: An array containing the names of the 7 characters.
  - `hp[]`: An array storing the endurance values corresponding to each character.

- `skill[]`: An array storing the skill values corresponding to each character.
- `conflictIndex`: The conflict index.
- `repairCost`: The ship repair cost.
- `cipherText`: A string representing the encrypted message.
- `resultText`: A decrypted string

- **Return value**: The string `plainText_TRUE` if the message is valid, or the string `plainText_FALSE` if the decoded result is invalid. If the message fails the checksum verification, the function returns an empty string.
- **Functionality**:
  The encrypted message has the format `message#XY`, where `XY` is a two-digit checksum.

  1. **Checksum verification:** compute

  $$checksum = \left(\sum ASCII(message[i])\right) \bmod 100$$

  If `checksum` is not equal to `XY`, the message is considered invalid.

  2. **Block-wise reversal:** determine

  $$key = (conflictIndex + repairCost) \bmod 26, \quad B = (key \bmod 5) + 4$$

  Split `message` into blocks of length B (the last block may be shorter) and reverse each block.

  3. **Reverse Caesar shift decoding:** for each character:
     - `A-Z`: shift backward by `key` using circular alphabet wrapping.
     - `a-z`: shift backward by `key` using circular alphabet wrapping.
     - `0-9`: shift backward by `key` mod `10` using circular digit wrapping.
     - Other characters remain unchanged.

4.4.2 Illustrative Example

Example 4.4

Assume:

$$conflictIndex = 120, \quad repairCost = 1500$$

Thus:

$$key = (120 + 1500) \bmod 26 = 8, \quad B = (8 \bmod 5) + 4 = 7$$

The received encrypted message is:

$$cipherText = \text{``AQ 7XK\#96''}$$

- **Step 1 (Checksum):** extract `message = ‘‘AQ 7XK’’` and the provided checksum `96`. Recompute:

  $$(65 + 81 + 32 + 55 + 88 + 75) \bmod 100 = 396 \bmod 100 = 96$$

  Therefore, the message is valid.
- **Step 2 (Block-wise reversal):** with $B = 7$, since the length of `message` is 6, there is only one block. Reverse this block:

  $$\text{``AQ 7XK''} \rightarrow \text{``KX7 QA''}$$

- **Step 3 (Reverse circular shift):** with $key = 8$:
  - `K` shifted backward by $8 \rightarrow$ `C`
  - `X` shifted backward by $8 \rightarrow$ `P`
  - `7` shifted backward by $(key \bmod 10) = 8 \rightarrow$ `9`
  - The space character remains unchanged
  - `Q` shifted backward by $8 \rightarrow$ `I`
  - `A` shifted backward by 8 (with wrapping) $\rightarrow$ `S`

  Therefore:

  $$plainText = \text{``CP9 IS''}$$

Since the decoded message contains the substring ‘‘CP9’’, it is considered valid, and the function returns `"CP9 IS_TRUE"`.

4.5 Task 5: Analyzing the Enies Lobby Map

After confirming the involvement of CP9, the Straw Hat crew decides to advance into Enies Lobby to rescue Robin. Enies Lobby is a heavily guarded judicial stronghold, featuring numerous dangerous zones and obstacles. To prepare for the infiltration, the crew must analyze the area map, which is modeled as a two-dimensional matrix.

Each cell in the matrix represents a region of Enies Lobby, with the following meanings:

- `-1`: Impassable area.
- `0`: Empty and safe area.
- `1`: Area guarded by regular soldiers.
- `2`: Area with watchtowers.
- `3`: Area controlled by CP9.

4.5.1 Task 5.1: Determining the Danger Threshold

- **Function name**: `analyzeDangerLimit`
- **Input parameters**:
  - `grid[][]`: A two-dimensional matrix representing the Enies Lobby map.
  - `rows`, `cols`: The number of rows and columns of the matrix.
- **Return value**: An integer value `dangerLimit` (the allowable danger threshold).
- **Functionality**:

  The function traverses the entire map to compute the following two quantities:

  1. **maxRowSum**: The maximum danger level across all rows. For each row, compute the sum of all non-negative cells (excluding cells with value `-1`), then take the maximum among these row sums.
  2. **maxCell**: The maximum value of a single cell in the entire matrix (representing the most dangerous area).

  The danger threshold is then determined as:

$$dangerLimit = maxRowSum + maxCell$$

4.5.2 Task 5.2: Evaluating the Access Route

- **Function name**: `evaluateRoute`
- **Input parameters**:

  - `grid[][]`: A two-dimensional matrix representing the Enies Lobby map.
  - `rows`, `cols`: The number of rows and columns of the matrix.
  - `dangerLimit`: The allowable danger threshold.

- **Return value**: A logical value (`true` or `false`).
- **Functionality**:

  The Straw Hat crew starts at cell `(0,0)` and must reach cell `(n-1,m-1)`. The crew is only allowed to move either to the right or downward.

  The function must perform the following:

  1. Check whether there exists at least one valid route from the starting cell to the destination cell without passing through any cells with value `-1`.
  2. If a valid route exists, compute the total danger level along that route.
  3. Compare the total danger level with `dangerLimit` to decide whether the crew should proceed into Enies Lobby.

  The function returns `true` if there exists a valid route and the total danger level does not exceed `dangerLimit`; otherwise, it returns `false`.

4.6 Conclusion

Through the tasks in Major Assignment 1, students progressively simulate a sequence of key events in the Water 7 arc, ranging from evaluating the condition of the Going Merry, the accumulation of internal conflict, to the investigation of CP9 and the analysis of the Enies Lobby map. The problems are designed with gradually increasing difficulty, requiring students to flexibly apply fundamental C++ programming concepts such as conditional statements, loops, one-dimensional arrays, two-dimensional arrays, string processing, and function design.

At the end of Major Assignment 1, the Straw Hat crew makes the decision to advance into Enies Lobby—a turning point that marks a direct confrontation with the World Government. However, what awaits them ahead are no longer tactical decisions, but real battles.

Wish you an enjoyable experience with the Major Assignment!!!

5 Requirements

To complete this major assignment, students must:

1. Carefully read the entire assignment description.

2. Download the file `initial.zip` and extract it. After extraction, students will obtain the following files: `main.cpp`, `main.h`, `water_seven.h`, `water_seven.cpp`, and sample input data files. Students are required to submit only two files: `water_seven.h` and `water_seven.cpp`. Therefore, **the file `main.h` must not be modified** when testing the program.
3. Students must use the following command to compile the program:
   g++ -o main main.cpp water_seven.cpp -I . -std=c++11
   Students must use the following command to run the program:
   ./main opw_tc_01_input
   The above commands are executed in a command prompt or terminal to compile and run the program. If students use an IDE to run the program, they must ensure that all required files are properly added to the project/workspace and that the IDE compilation command is correctly configured. IDEs typically provide *Build* and *Run* buttons; however, the default build configuration usually compiles only `main.cpp`. Students must configure the IDE to modify the compilation command by adding `water_seven.cpp` and the options `-std=c++11` and `-I ..`
4. The program will be graded on a Unix-based platform. The grading environment and compiler may differ from the student's local environment. The submission system on LMS is configured to closely match the actual grading environment. Students must test their program on the LMS submission system and fix all errors encountered there to ensure correct results during official grading.
5. Modify the files `water_seven.h` and `water_seven.cpp` to complete this major assignment and ensure the following requirements are met:
   - There must be exactly one **#include** directive in `water_seven.h`:

     #include "main.h"

   - In `water_seven.cpp`, there must be exactly one **#include** directive:

     #include "water_seven.h"

   - Apart from the two includes above, **no other #include directives are allowed** in these files (including #include directives that have been commented out)
   - Implement all functions described in the tasks of this major assignment.
6. Students are encouraged to implement additional helper functions to complete this assignment.

6 Submission

Students must submit only two files: `water_seven.h` and `water_seven.cpp`, before the deadline specified in the link "`Assignment 1 - Submission`". A number of basic test cases will be used to verify that the submitted program can be successfully compiled and executed. Students may submit multiple times; however, only the **final submission** will be graded.

Due to system load limitations when many students submit simultaneously, students are strongly encouraged to submit their work as early as possible. Students bear full responsibility for any risks associated with late submission (within one hour before the deadline). Once the submission deadline has passed, the system will be closed and no further submissions will be accepted. Submissions via any other methods will not be considered.

7 Additional Regulations

1. Students must complete this assignment on their own and prevent others from copying their work. Failure to do so may result in penalties under the university's academic integrity policies.
2. All grading decisions made by the course instructors are final.
3. Students will not be provided with test cases after grading.
4. The content of this assignment is aligned with some questions in the final exam.
5. If students use code from AI-generated programs or automated code generators without understanding how the code works, their assignment will receive a score of 0.

8 Harmony for the Assignment

Some questions in the final exam will be directly related to this assignment.

Students must complete this assignment by themselves. If a student cheats in this assignment, they will be unable to answer related Harmony questions in the final exam and will receive a score of 0 for the assignment.

Students **must** answer the Harmony questions in the final exam. Failure to do so will result in a score of 0 for the assignment and failure in the course. **No exceptions or justifications will be accepted.**

9 Academic Integrity Policy

This assignment must be completed independently. Students will be considered to have committed academic misconduct if:

- There is an unusual similarity between submitted code. In such cases, **all involved submissions** will be considered as cheating. Therefore, students must protect their code from being copied.
- A student submits another student's work under their own account.
- A student does not understand the code they submitted, except for the starter code provided in the initial setup. Students may refer to any resources but must fully understand all the code they submit. If a student does not fully understand the source code they are referencing, they are explicitly warned **not to use it** and should instead rely on what they have learned to write the program.
- A student accidentally submits another student's work using their own account.
- A student uses code generated by automated tools without understanding its meaning.

If a student is found guilty of academic misconduct, they will receive a score of 0 for the entire course (not just this assignment).

NO EXCEPTIONS WILL BE ACCEPTED, AND NO JUSTIFICATIONS WILL BE CONSIDERED!

After each assignment submission, some students will be randomly selected for an interview to verify that they personally completed their submitted work.

10 Changes from the Previous Version

References