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

THE GENERAL OFFENSIVE AND UPRISING SPRING 1975 Part 1: Tay Nguyen Campaign

Version 1.0

Ho Chi Minh City, 02/2025

ASSIGNMENT SPECIFICATION

Version 1.0

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

2 Introduction

This assignment is inspired by the 50th anniversary of the success of the 1975 Spring General Offensive and Uprising, which led to the complete liberation of South Vietnam and national reunification.

After the Paris Peace Accords in 1973, the United States withdrew its troops from South Vietnam but continued to provide military aid to the Saigon government. The U.S. imperialists aimed to use the Army of the Republic of Vietnam (ARVN) to implement neocolonialism in South Vietnam, turning the region into a new-style colony. In July 1973, the 21st Central Party Conference reaffirmed that the revolutionary path for South Vietnam remained a path of armed struggle and outlined major tasks for both the North and South. By July 1974, the Party directed the General Staff of the People's Army of Vietnam to develop a strategic plan to liberate South Vietnam within two years (1975-1976), with the possibility of accelerating liberation in 1975 if favorable conditions arose [2]. By late 1974 and early 1975, the Party assessed that the opportunity for liberation had arrived.

The Spring 1975 General Offensive and Uprising commenced on March 4th, featuring three major military campaigns. The first and decisive campaign was the historic Tay Nguyen Campaign, which played a crucial role in paving the way for the Hue-Da Nang Campaign and the final Ho Chi Minh Campaign [3].

The objective of the Tay Nguyen Campaign was to eliminate a significant portion of enemy forces, liberate the southern provinces of the Central Highlands (Đắk Lắk, Phú Bổn, Quảng

Dức), execute strategic division, and establish a new strategic position for the overall battlefield in South Vietnam [1].

In just one month (March 1975), the Tay Nguyen Campaign achieved a decisive victory. In this first assignment, inspired by the events of the campaign, students will use programming to simulate key aspects of its historical progression.

3 Input Data

3.1 Description of Input Data

The input data for the program is stored in a file, with the filename assigned to the variable **file_input**. This file contains the following information:

```
[LF1_0,LF1_1,...,LF1_16]
[LF2_0,LF2_1,...,LF2_16]
EXP1␣EXP2
T1␣T2
E
```

Where:

- **LF_1 and LF_2**: Each value in these two arrays represents the number of soldiers at each rank in two key military corps during the Tay Nguyen Campaign. Both arrays contain 17 elements, each corresponding to a rank from the lowest to the highest: **Private, Private First Class, Corporal, Sergeant, Staff Sergeant, Second Lieutenant, First Lieutenant, Captain, Major, Lieutenant Colonel, Colonel, Senior Colonel, Brigadier General, Major General, Lieutenant General, General.**
  - $[LF1_0, LF1_1, ..., LF1_{16}]$: Number of soldiers at each rank in Military Corps 1.
  - $[LF2_0, LF2_1, ..., LF2_{16}]$: Number of soldiers at each rank in Military Corps 2.

  Each element in the array is an integer within the range $[0, 1000]$. If the total force exceeds 1000, it must be reset to 1000. If it falls below 0, it must be reset to 0.
- **EXP_1 and EXP_2**: These represent the combat experience levels of our military units in the two corps. Experience increases with battles and intelligence gathering. This value is an integer within the range $[0, 600]$. If it exceeds 600, it must be reset to 600. If it is below 0, it must be reset to 0.

- **T_1 and T_2**: These represent the supply resources, including food, weapons, and ammunition for the two military corps. This value is an integer within the range $[0, 3000]$. If it exceeds 3000, it must be reset to 3000. If it is below 0, it must be reset to 0.
- **E**: E represents the event code for the missions in this assignment, which is an integer in the range $[0, 99]$.

Example:

```
[100,80,60,40,20,10,5,2,1,0,0,0,0,0,0,0,0]
[120,90,70,50,30,15,10,5,3,2,1,0,0,0,0,0,0]
400␣350
2500␣2800
99
```

3.2 Task 0: Reading Input Data (1.0 point)

Before executing other tasks in the Tay Nguyen campaign, the army must gather and process information regarding the available forces, combat experience, supply resources, and impacting events. The first task for students is to implement a function that reads and processes data from the input file.

3.2.1 Function Description

- **Function Name**: `readFile`
- **Input Parameters**:
  - **filename**: The name of the file containing the input data.
  - **LF1[17]**: A 1D array storing the number of soldiers at each rank in Army Corps 1.
  - **LF2[17]**: A 1D array storing the number of soldiers at each rank in Army Corps 2.
  - **EXP1, EXP2**: The combat experience levels of the two corps.
  - **T1, T2**: The supply resources of the two corps.
  - **E**: The event code that affects the mission.
- **Return Value**:
  - `true`: If the file is successfully opened and read.
  - `false`: If the file does not exist or an error occurs while reading the data.

3.2.2 Detailed Description

Students are required to implement the `readFile` function to read content from the input file, store it in a character array before parsing and assigning values to the corresponding variables. Specific requirements include:

- Read the entire content of the input file and store each line in the two-dimensional character array `char data[MAX_LINES][MAX_LINE_LENGTH]`.
- Students must extract data from the character array and assign the values to the variables `LF1`, `LF2`, `EXP1`, `EXP2`, `T1`, `T2`, and `E`.
- If the number of data lines is insufficient or the format is incorrect, the function must return `false`.

3.2.3 Example

Assume the input file contains the following data:

```
[100,80,60,40,20,10,5,2,1,0,0,0,0,0,0,0,0]
[120,90,70,50,30,15,10,5,3,2,1,0,0,0,0,0,0]
400␣350
2500␣2800
99
```

After reading the file and processing the data, the `readFile` function must correctly assign the values as follows:

- **LF1**: `[100,80,60,40,20,10,5,2,1,0,0,0,0,0,0,0,0]`
- **LF2**: `[120,90,70,50,30,15,10,5,3,2,1,0,0,0,0,0,0]`
- **EXP1**: `400`, **EXP2**: `350`
- **T1**: `2500`, **T2**: `2800`
- **E**: `99`

4 Tasks

During the Tay Nguyen Campaign, our army carried out five critical missions to achieve a complete victory. Students are required to develop a simulation program to replicate some of the key operations in the battle and liberation of Tay Nguyen. The specific tasks are described as follows:

4.1 Task 1: Force Assembly (1.0 point)

Before launching the Tay Nguyen campaign, our army needed to gather sufficient troops in two key army corps. In this mission, the military strength of each corps is determined based on the number of soldiers at various ranks. Each rank contributes a corresponding strength value to the total military force strength **LF** according to a weighted formula.

4.1.1 Function Description

- **Function Name**: **gatherForces**
- **Input Parameters**:
  - **LF1[17]**: A 1D array consisting of 17 elements representing the number of soldiers at each rank in Army Corps 1.
  - **LF2[17]**: A 1D array consisting of 17 elements representing the number of soldiers at each rank in Army Corps 2.
- **Return Value**:
  - The function returns the **Total Military Force Strength**.

4.1.2 Detailed Description

The value of **Total Military Force Strength** is determined by the number of soldiers at each rank, with each rank having a specific strength weight, reflecting its contribution to the overall military force strength. The specific weight for each rank is defined as follows:

Index	Rank	Weight	Role and Description
0	Private	1	Newly enlisted soldier, minimal combat experience.
1	Private First Class	2	More service time and higher field experience than a Private.
2	Corporal	3	Leads small groups, capable of guiding lower-ranked soldiers.
3	Sergeant	4	Experienced, assists in leading a platoon.
4	Staff Sergeant	5	Supports company-level leadership, may take temporary command.
5	Second Lieutenant	7	The lowest-ranking officer, commands a platoon, crucial in small battles.
6	First Lieutenant	8	Commands a platoon or small company, skilled in tactical organization.
7	Captain	9	Holds significant leadership roles within a company unit.
8	Major	10	Leads a company, participates in battalion-level planning.
9	Lieutenant Colonel	12	Commands a battalion, responsible for tactical operations.
10	Colonel	15	Leads a regiment or serves as a high-ranking staff officer.
11	Senior Colonel	18	Plays a critical role in commanding a regiment or division.
12	Brigadier General	20	Commands a division, involved in regional military strategy.
13	Major General	30	Leads a corps, with authority over large forces in campaigns.
14	Lieutenant General	40	Commands a military region, responsible for major campaigns.
15	General	50	One of the highest-ranking commanders, involved in national military leadership.
16	General of the Army	70	Supreme commander, holds the ultimate decision-making power in military campaigns.

Table 1: Military Force Strength by Rank

Total Military Force Strength for each corps is calculated using the formula:

$$\mathbf{LF_1} = \sum_{i=0}^{16}(\text{LF1}[i] \times \text{weight}[i])$$

$$\mathbf{LF_2} = \sum_{i=0}^{16}(\text{LF2}[i] \times \text{weight}[i])$$

From this, the **Total Military Force Strength** of both corps is determined as follows:

$$\mathbf{LF} = \mathbf{LF_1} + \mathbf{LF_2}$$

4.1.3 Example

Example 4.1

With the given input data:

```
[200,150,100,80,50,30,20,10,5,2,1,1,1,1,0,0,0]
[250,200,150,100,80,50,30,20,10,5,2,1,1,1,1,0,0]
300␣300
3000␣3200
99
```

Military force strength of Corps 1:

$$\mathbf{LF_1} = (200 \times 1) + (150 \times 2) + (100 \times 3) + ... + (0 \times 70) = 1000$$

Military force strength of Corps 2:

$$\mathbf{LF_2} = (250 \times 1) + (200 \times 2) + (150 \times 3) + ... + (0 \times 70) = 1000$$

Total military force strength: $\mathbf{LF} = 1000 + 1000 = 2000$

4.2 Task 2: Decoy Operations (2.0 points)

In this campaign, our army employed the **decoy strategy** to mislead the enemy, allowing us to capture key targets without facing strong resistance from opposing forces.

To divert the enemy forces, our troops created false signals, leading them to believe that the main attack would take place in the **northern Central Highlands**, while in reality, the primary target was in the **southern Central Highlands - Buon Ma Thuot**.

Key Decoy Locations [1]:

- **Kon Tum**:
  - Our forces staged false troop movements to make the enemy believe we were preparing to attack Kon Tum.
  - Radio signals were transmitted to simulate communication for an impending assault.
- **Pleiku**:
  - The army constructed fortifications and transported fake supplies to convince the enemy of an imminent large-scale attack on Pleiku.
  - Information about the attack was spread via local militia and propaganda.
- **Gia Lai (Thanh An, Route 19)**:
  - Division 968 was ordered to march from southern Laos to the Central Highlands, replacing the main forces of Divisions 320 and 10 to conduct decoy operations in Kon Tum - Pleiku.
  - Artillery bombardments were carried out in Pleiku to pin down enemy forces.

The most important objective of the campaign was **Buon Ma Thuot**, as it served as the strategic defense center of the Saigon government's army in the Central Highlands. Our forces concentrated a significant portion of their strength to capture this area.

Key Targets to Capture [1]**:**

- **Buon Ma Thuot (most critical target)**:
  - The largest administrative, military, and logistics center of the Saigon government in the Central Highlands.
  - Capturing Buon Ma Thuot would lead to the collapse of the entire enemy defense system in the Central Highlands.
- **Duc Lap (west of Buon Ma Thuot)**:
  - A key defensive stronghold guarding the western gateway to Buon Ma Thuot.
  - Attacked first by Division 10 to cut off reinforcements from this direction.
- **Dak Lak (surrounding Buon Ma Thuot)**:
  - Cut off the supply lines of the Saigon army from the south and east of Buon Ma Thuot.

  - Isolate enemy forces in the town of Buon Ma Thuot.

- **National Route 21 and National Route 14 (securing the Central Highlands)**:
  - On March 4, 1975, Regiment 25 blocked National Route 21 from Dak Lak to Khanh Hoa, preventing enemy retreat.
  - From March 6 to 8, 1975, Division 320 blocked National Route 14 from Pleiku to Buon Ma Thuot, isolating the battlefield.

- **Hoa Binh Airport, Mai Hac De Storage Facility**:
  - Attacked at 2:00 AM on March 10, 1975, to cut off enemy air reinforcements.

Based on the above information, we have the following data tables:

ID	Decoy Target	English Name
0	Kon Tum	Kon Tum
1	Pleiku	Pleiku
2	Gia Lai	Gia Lai

Table 2: List of Decoy Targets

ID	Primary Targets to Capture	English Name
3	Buon Ma Thuot	Buon Ma Thuot
4	Duc Lap	Duc Lap
5	Dak Lak	Dak Lak
6	National Route 21	National Route 21
7	National Route 14	National Route 14

Table 3: List of Primary Targets to Capture

4.2.1 Task 2.1: Decoy Strategy

In this task, we will implement the **decoy strategy** to mislead the enemy about the actual target of our army's attack. Specifically, the decoy strategy is executed by **encoding** the information about the target to be captured, causing the enemy to misunderstand our true direction of attack. The art of military deception is a vast topic; in this assignment, we assume simple encryption methods suitable for the scope of this task.

Rules for Encoding Decoy Information:

- Each decoy target is represented as a string consisting of:

- **The English name of the decoy target**.
- Between 1 and 3 integer values encoding the **ID of the actual target**.

- The integers encoding the ID can appear at **any position** in the string. If there are 2 or 3 numbers, they will not be adjacent.
- Each number in the string is a positive integer in the range $[0, 100]$.

Rules for Decoding the ID:

- If there is exactly **1 integer**, that integer is the **ID of the actual target**.
- If there are exactly **2 integers**, the target ID is computed as:

$$\text{ID} = [(x_1 + x_2) \mod 5] + 3$$

- If there are exactly **3 integers**, the actual target ID is determined based on the **largest** of the three numbers, using the formula:

$$\text{ID} = [(\max(x_1, x_2, x_3)) \mod 5] + 3$$

- If no numbers are found or more than 3 numbers are present, the information is invalid, and the function returns `"INVALID"`.

Function Description

- **Function Name**: `determineRightTarget`
- **Input Parameters**:
  - **target**: A string containing the decoy target name along with the encoded ID of the actual target.
- **Assumption**: The ID is an integer within the range $[0, 100]$.
- **Return Value**:
  - The English name of the actual target to be captured.
  - If no valid ID is found, the function returns `"INVALID"`.

Examples

Example 4.2

Case 1: Exactly 1 Number

The input string contains exactly one number. The ID of the actual target is that number.

```
Kon3␣Tum
```

Decoding result:

- `Kon3 Tum` ⇒ The enemy believes the attack target is **Kon Tum**, but in reality, the main objective is **Buon Ma Thuot** (ID = 3).
- The function returns **"Buon Ma Thuot"**.

Example 4.3

Case 2: Exactly 2 Numbers

The input string contains two numbers. The ID is computed using the formula:

$$\text{ID} = [(x_1 + x_2) \mod 5] + 3$$

```
Ple9i␣ku5
```

Computing the ID:

$$\text{ID} = [(9 + 5) \mod 5] + 3 = [14 \mod 5] + 3 = 4 + 3 = 7$$

Decoding result:

- `Ple9iku5` ⇒ The enemy believes the attack target is **Pleiku**, but in reality, the main objective is **National Route 14** (ID = 7).
- The function returns **"National Route 14"**.

Example 4.4

Case 3: Exactly 3 Numbers

The input string contains three numbers. The ID is calculated using the formula:

$$\text{ID} = [(\max(x_1, x_2, x_3)) \mod 5] + 3$$

```
Ple1Ku4Nat9
```

Computing the ID:

$$\text{ID} = [(\max(1, 4, 9)) \mod 5] + 3 = [9 \mod 5] + 3 = 4 + 3 = 7$$

Decoding result:

- `Ple1Ku4Nat9` $\Rightarrow$ The enemy believes the attack target is **Pleiku**, but in reality, the main objective is **National Route 14** (ID $= 7$).
- The function returns **"National Route 14"**.

Example 4.5

Case 4: No Valid Number or More than 3 Numbers

If the input string contains no numbers or more than three numbers, the result will be "INVALID".

```
KonTum
```

Decoding result:

- No valid numbers are found in the string.
- The function returns **"INVALID"**.

```
Buon1Ma3Thuot5-8
```

Decoding result:

- The string contains four numbers.
- The function returns **"INVALID"**.

4.2.2 Task 2.2: Decoding

To ensure secrecy and unpredictability, information about the next attack target of our army is encrypted in classified messages. Therefore, one of the crucial tasks is to decrypt these messages to accurately determine the next target.

- **Function Name**: `decodeTarget`
- **Input Parameters**:
  - **message**: An encrypted string containing information about the primary target.
  - **EXP1**: Intelligence experience of our forces in Army Corps 1.
  - **EXP2**: Intelligence experience of our forces in Army Corps 2.
- **Function Requirements**:
  - Decode the **message** using one of two methods: **Caesar Cipher** or **String Reversal**.
  - After decoding, check if the resulting string matches any entry in the **List of Primary Targets**.
  - If a match is found, return the successfully decoded target name. Otherwise, return `"INVALID"`.

Rules for Selecting the Decoding Method:

- If $\mathbf{EXP_1} \geq 300$ and $\mathbf{EXP_2} \geq 300$, use the **Caesar Cipher** method.
- If $\mathbf{EXP_1} < 300$ or $\mathbf{EXP_2} < 300$, use the **String Reversal** method.

Decoding Methods:

1. **Caesar Cipher**:
   - Each character in the message is shifted by $(\mathbf{EXP_1} + \mathbf{EXP_2}) \mod 26$ positions in the alphabet (excluding spaces).
   - Character transformation formula:

$$\text{New Character} = (\text{Old Character} - \text{shift} + 26) \mod 26$$

   - If the resulting character is outside the alphabetical range (A-Z, a-z), keep it unchanged.
2. **String Reversal**: Reverse the entire input string.

Examples:

Example 4.6

Case 1: Using Caesar Cipher

Input Data:

- **EXP1** = 350, **EXP2** = 400
- Encrypted string: "Zwg Hwg"

Decoding Steps:

$$\text{Shift left} = (350 + 400) \mod 26 = 750 \mod 26 = 22$$

Shifting each character to the left (excluding spaces):

$$\text{"Z"} \rightarrow \text{"D"}$$
$$\text{"w"} \rightarrow \text{"a"}$$
$$\text{"g"} \rightarrow \text{"k"}$$
$$\text{" "} \rightarrow \text{" "}$$
$$\text{"H"} \rightarrow \text{"L"}$$
$$\text{"w"} \rightarrow \text{"a"}$$
$$\text{"g"} \rightarrow \text{"k"}$$

Decoded Result: "Dak Lak"

Cross-checking with target list: Found in the list of primary targets ⇒ Function returns `"Dak Lak"`.

Example 4.7

Case 2: Using String Reversal

Input Data:

- **EXP1** $= 150$, **EXP2** $= 320$
- Encrypted string: `"Pal cUd"`

Decoding Steps:

$$\text{"Pal cUd"} \rightarrow \text{"dUc laP"}$$

Cross-checking with target list: Found in the list of primary targets ⇒ Function returns `"Duc Lap"`.

4.3 Task 3: Logistics Management (2.0 points)

During the Tay Nguyen Campaign, maintaining **logistical supplies**, including food, weapons, and ammunition, played a crucial role in ensuring the combat effectiveness of our army. To efficiently manage the supply distribution among military corps, the army needs to allocate resources appropriately based on actual demand and consumption in battle.

The military must implement a logistics management system to monitor the available supplies at **two corps** and adjust distribution based on three key factors:

- **Military forces** ($\mathbf{LF_1}, \mathbf{LF_2}$): A corps with a larger force requires more supplies.
- **Combat experience level** ($\mathbf{EXP_1}, \mathbf{EXP_2}$): Units with higher experience can utilize resources more effectively.
- **Historical events** ($\mathbf{E}$): Certain events may alter the supply levels of each corps.

4.3.1 Function Description

- **Function name**: `manageLogistics`
- **Input parameters**:
  - $\mathbf{LF_1}$, $\mathbf{LF_2}$: The number of troops in the two corps.
  - $\mathbf{EXP_1}$, $\mathbf{EXP_2}$: The experience level of the two corps.
  - $\mathbf{T_1}$, $\mathbf{T_2}$: The current amount of supplies in the two corps (**passed by reference**, allowing value updates).
  - $\mathbf{E}$: The historical event code affecting logistics.

- **Return value**:
  - The function does not return a value but directly updates the two variables $\mathbf{T_1}$ and $\mathbf{T_2}$ after adjustments.
  - All values of $\mathbf{T_1}, \mathbf{T_2}$ must be within the range $[0, 3000]$. If any calculation results in a non-integer value, it must be immediately rounded up.

4.3.2 Detailed Description

Supply distribution is adjusted using the following formula:

$$\Delta T_1 = \left(\frac{\mathbf{LF_1}}{\mathbf{LF_1} + \mathbf{LF_2}} \times (\mathbf{T_1} + \mathbf{T_2})\right) \times \left(1 + \frac{\mathbf{EXP_1} - \mathbf{EXP_2}}{100}\right)$$

$$\Delta T_2 = (\mathbf{T_1} + \mathbf{T_2}) - \Delta T_1$$

Historical Event-Based Adjustments: The historical event code affecting logistics is categorized as follows:

- **If** $\mathbf{E} = 0$: No special events, apply the standard formula above.
- **If** $\mathbf{E} \in [1, 9]$:
  - Corps 1 loses $(\mathbf{E} \times 1\%)$ of its supplies.
  - Corps 2 loses $(\mathbf{E} \times 0.5\%)$ of its supplies.
- **If** $\mathbf{E} \in [10, 29]$:
  - Each corps gains an additional $(\mathbf{E} \times 50)$ supply units.
- **If** $\mathbf{E} \in [30, 59]$:
  - Corps 1 receives an increase of $(\mathbf{E} \times 0.5\%)$ in supplies.
  - Corps 2 receives an increase of $(\mathbf{E} \times 0.2\%)$ in supplies.
- **If** $\mathbf{E} \in [60, 99]$: Supply routes are disrupted, and no adjustments are made to supplies.

4.3.3 Examples

Example 4.8

Case 1: Event Reducing Supplies

```
[250,200,150,100,80,50,30,20,10,5,2,1,1,1,1,0,0]
[300,250,200,150,100,80,50,30,20,10,5,2,1,1,1,0,0]
400␣450
2800␣3000
5
```

Since **E** $= 5$ falls within the range $[1, 9]$, supplies are reduced:

$$\mathbf{T_1} = 2800 - (2800 \times 5\%) = 2660$$

$$\mathbf{T_2} = 3000 - (3000 \times 2.5\%) = 2925$$

Example 4.9

Case 2: Event Increasing Logistics Support

```
[200,180,150,120,90,60,40,30,20,10,5,2,1,1,0,0,0]
[220,190,160,130,100,70,50,30,20,10,5,2,1,1,0,0,0]
350␣380
2600␣2800
20
```

Since **E** $= 20$ falls within the range $[10, 29]$, each corps receives an additional $20 \times 50 = 1000$ units:

$$\mathbf{T_1} = \min(2600 + 1000, 3000) = 3000$$

$$\mathbf{T_2} = \min(2800 + 1000, 3000) = 3000$$

4.4 Task 4: Attack Planning (2.0 points)

During the Tay Nguyen Campaign, our army needs to formulate a strategic attack plan to eliminate all enemy strongholds in key areas. This plan is based on the combined strength of our forces and the enemy presence at each position on the battlefield. Careful planning will help optimize resources and ensure victory with minimal losses.

4.4.1 Task Description

The army must determine its chances of victory through a series of analyses and calculations:

The battlefield is modeled as a **combat matrix** of size 10×10. Each cell in the matrix represents a combat zone, with a numerical value indicating the enemy stronghold's strength at that position.

To achieve a complete victory, the army must eliminate all enemy strongholds across the battlefield, ensuring that no enemy position remains.

To neutralize each enemy stronghold, our army must expend a corresponding amount of strength. The required strength varies depending on whether the stronghold is located in an even or odd-numbered row of the matrix.

The **combined strength** of our army is determined based on three key factors:

- Number of troops (**LF**): Reflects the scale of forces.
- Combat experience (**EXP**): Represents combat capability.
- Supplies (**T**): Ensures sustainability in battle.

The attack simulation proceeds by iterating through each cell in the combat matrix. At each position, our army's strength is reduced by an amount corresponding to the strength of the enemy stronghold in that position, according to different formulas for even and odd rows.

Once all strongholds have been engaged, the final outcome is determined based on the remaining strength of our forces:

- If positive: The army has sufficient strength to secure victory.
- If negative: Additional supplies are required to ensure success.

4.4.2 Function Description

- **Function Name**: `planAttack`

- **Input Parameters**:
  - **$\mathbf{LF_1}, \mathbf{LF_2}$**: The number of troops in Corps 1 and Corps 2.
  - **$\mathbf{EXP_1}, \mathbf{EXP_2}$**: The combat experience of Corps 1 and Corps 2.
  - **$\mathbf{T_1}, \mathbf{T_2}$**: The available supplies for Corps 1 and Corps 2.
  - **battleField**[**10**][**10**]: A matrix representing enemy strength at each position.
- **Processing Steps**:
  - Compute the initial combined strength:

$$S = (LF_1 + LF_2) + (EXP_1 + EXP_2) \times 5 + (T_1 + T_2) \times 2$$

  - Iterate through the battlefield matrix and update S for each position (i, j):
    * Odd rows:

$$S = S - \left(\frac{\textbf{battleField}_{i,j} \times 3}{2} \right)$$

    * Even rows:

$$S = S - \left(\frac{\textbf{battleField}_{i,j} \times 2}{3} \right)$$

- **Return Value**:
  - The final combined strength S after processing the entire battlefield matrix.
  - **Note:** The returned value must be rounded up to the nearest integer.
- **Outcome Interpretation**:
  - If S is positive: Our army has overwhelming strength, ensuring victory.
  - If S is negative: The absolute value of S represents the amount of additional supplies required to secure victory.

4.4.3 Examples

Example 4.10

Case 1: Our Army Wins

Given Input:

- $\mathbf{LF_1} = 300$, $\mathbf{LF_2} = 280$
- $\mathbf{EXP_1} = 450$, $\mathbf{EXP_2} = 470$
- $\mathbf{T_1} = 2500$, $\mathbf{T_2} = 2600$
- Battlefield matrix:

106	15	20	25	305	635	540	145	50	55
25	18	24	330	36	442	48	54	660	665
14	21	28	35	452	49	56	63	70	77
162	24	323	404	484	60	40	72	80	88
181	27	36	52	543	63	72	81	90	99
5	30	40	501	602	70	80	90	100	110
22	33	442	55	66	77	58	99	10	121
24	36	48	60	72	84	96	108	20	132
264	39	525	65	78	91	104	70	130	143
28	42	56	50	84	98	125	126	140	154

Initial Strength Calculation:

$$S = (300 + 280) + (450 + 470) \times 5 + (2500 + 2600) \times 2 = 15380$$

After processing the battlefield matrix:

$$S = 193$$

Since $S > 0$, our army achieves victory.

The function returns 193.

Example 4.11

Case 2: Additional Supplies Needed

Given Input:

- $\mathbf{LF_1} = 300$, $\mathbf{LF_2} = 280$
- $\mathbf{EXP_1} = 450$, $\mathbf{EXP_2} = 470$
- $\mathbf{T_1} = 2500$, $\mathbf{T_2} = 2600$
- Battlefield matrix:

506	15	20	25	305	635	540	145	50	55
325	18	24	330	36	442	48	54	660	665
14	21	28	35	452	49	56	63	70	77
562	24	323	404	484	60	40	72	80	88
181	27	36	52	543	63	72	81	90	99
505	30	40	501	602	70	80	90	100	110
922	33	442	55	66	77	58	99	110	121
124	36	48	60	72	84	96	108	120	132
264	39	525	65	78	91	104	170	130	143
28	42	56	50	84	98	125	126	140	154

Initial Strength Calculation:

$$S = (300 + 280) + (450 + 470) \times 5 + (2500 + 2600) \times 2 = 15380$$

After processing the battlefield matrix:

$$S = -2908$$

Since $S < 0$, our army requires an additional 2908 units of supplies to secure victory.
The function returns -2908.

4.5 Task 5: Supply Reinforcement (2.0 points)

After completing Task 4, if our army does not have enough strength to secure victory, this task will focus on obtaining additional supplies from the rear to ensure combat capability.

4.5.1 Task Description

The additional supplies are stored in a **5x5 matrix**, where each cell contains a certain amount of supplies. The total supply in this matrix does not exceed 3000, and our army will not have a shortage of more than 3000 supply units.

According to historical events, our army is guaranteed to win. Therefore, **in all cases**, the supply matrix will always contain at least one set of 5 cells with a total value greater than the required additional supply.

Our task is to select exactly **5 supply packages** (i.e., 5 cells in the matrix) such that:

- The total of the 5 selected supply packages is **greater than** the required supply (**shortfall**).
- The total amount of supply from the 5 selected cells must be **as small as possible** to optimize resource usage.

4.5.2 Function Description

- **Function Name**: `resupply`
- **Input Parameters**:
  - **shortfall**: The amount of supply needed to be replenished.
  - **supply[5][5]**: A 5×5 matrix containing the supply amount at each position.
- **Processing Steps**:
  1. Traverse the **supply[5][5]** matrix to identify all possible supply sources.
  2. Select exactly **5 cells** such that their total is **greater than or equal to** the required supply (**shortfall**).
  3. If multiple combinations satisfy the condition, select the combination with the **smallest possible total supply**.
  4. For optimization, all 25 cells can be sorted in ascending order before selecting the optimal combination.
- **Precondition:** The supply matrix always contains at least one valid combination.
- **Return Value**:
  - The total value of the 5 selected cells. If multiple combinations have the same minimum total value, any of them can be returned.

4.5.3 Example

Example 4.12

Case 1: Finding the Optimal Combination

Given Input:

- **shortfall** $= 1050$
- **Supply Matrix**:

150	200	180	90	110
70	80	120	140	160
220	240	200	190	130
100	110	300	280	320
170	210	260	230	290

Selecting the 5 Optimal Cells:

$$(0,1) = 200, \quad (0,2) = 180, \quad (2,0) = 220, \quad (3,2) = 300, \quad (4,0) = 170$$

Total Supply Obtained:

$$200 + 180 + 220 + 300 + 170 = 1070$$

Since $1070 \geq 1050$ and this is the smallest possible total that meets the requirement, this is the optimal selection.

Function Returns: 1070.

4.6 Conclusion

After thorough preparation, our army has successfully assembled forces, employed decoy tactics to mislead the enemy, ensured stable logistics, and developed a strategic attack plan. Each task in the Tay Nguyen Campaign has been systematically executed, demonstrating the coordinated efforts between army corps and the unwavering determination to achieve victory.

However, the general offensive is not yet over. After liberating Tay Nguyen, the battlefield situation in the South changed rapidly. The enemy fell into chaos, retreating from key positions, creating an opportunity for our army to continue the offensive and completely liberate the South. To achieve the final objective, our army must launch the decisive campaign: the Ho Chi

Minh Campaign.

Will our army be able to maintain its strength, break through the enemy's final defensive lines, and advance into Saigon? Let's continue executing the crucial tasks in Assignment 2.

Enjoy working on your Assignment!

5 Requirements

To complete this assignment, students must:

1. Read this entire specification document.
2. Download the file initial.zip and extract it. After extraction, students will receive the following files: main.cpp, main.h, tay_nguyen_campaign.h, tay_nguyen_campaign.cpp, and sample data files. Students must submit only the two files tay_nguyen_campaign.h and tay_nguyen_campaign.cpp and must not modify main.h when testing the program.
3. Students must use the following command to compile the program:
   g++ -o main main.cpp tay_nguyen_campaign.cpp -I . -std=c++11
   Students must use the following command to run the program:
   ./main tnc_tc_01_input
   These commands should be used in the command prompt/terminal to compile and run the program. If students use an IDE to run the program, they should ensure that all files are added to the project/workspace and modify the compilation command accordingly. Most IDEs provide buttons for "Build" and "Run." When clicking "Build," the IDE executes a compilation command, usually only compiling main.cpp. Students must configure the IDE to include tay_nguyen_campaign.cpp, add the `-std=c++11` and `-I .` options.
4. The program will be graded on a Unix-based platform. The grading environment and the student's compiler setup may differ. The submission system on BKeL is configured to match the grading environment. Students must test their programs on the submission platform and fix all errors occurring there to ensure correct results during actual grading.
5. Modify only the files tay_nguyen_campaign.h and tay_nguyen_campaign.cpp to complete this assignment while ensuring the following conditions:

   - The `tay_nguyen_campaign.h` file must contain only one **#include** directive:

     #include "main.h"

   - The `tay_nguyen_campaign.cpp` file must contain only one **#include** directive:

#include "tay_nguyen_campaign.h"

- Apart from the two includes mentioned above, no other **#include** directives are allowed in these files.
- Implement all functions described in this assignment.

6. Students are encouraged to write additional functions to complete this assignment.

6 Submission

Students must submit only two files: `tay_nguyen_campaign.h` and `tay_nguyen_campaign.cpp`, before the deadline specified in the "Assignment 1 - Submission" section. Some simple test cases will be used to check students' submissions to ensure that the program compiles and runs correctly. Students can submit their work as many times as they want, but only the final submission will be graded.

Since the system may experience heavy traffic near the deadline, students are advised to submit their work as early as possible. Students bear the risk if they submit their work too close to the deadline (within 1 hour before the deadline). Once the deadline has passed, the system will close, and no further submissions will be accepted. Submissions via any other method will not be accepted.

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

[1] Phương Hà. Cuộc nghi binh lừa địch hoàn hảo trong Chiến dịch Tây Nguyên. *Báo Tin tức*, 2017. Accessed on 07/02/2025.

[2] Nguyễn Thành Hữu. Chiến dịch Tây Nguyên trong lịch sử Quân đội nhân dân Việt Nam. *Cổng thông tin điện tử Bộ Quốc phòng nước Cộng hòa Xã hội Chủ nghĩa Việt Nam*, 2013. Accessed on 07/02/2025.

[3] Đặng Cường. Ngày 4-3-1975: Quân đội nhân dân Việt Nam tiến hành Chiến dịch Tây Nguyên. *Báo Quân đội Nhân dân*, 2022. Accessed on 07/02/2025.