EXAM PREPARATION

Problem Solving (2.5 hours)

Problem 1: School Textbook Distribution Problem

Imagine you are helping a school distribute books among classrooms for a new semester. The school has a fixed total number of books available, for example, 720 books. Ideally, each classroom should receive the same number of books to ensure fair distribution among students. However, due to a recent software update, the inventory management system has temporarily disabled division (/) and modulo (MOD) operations. The task is to determine how many classrooms can receive the same number of books, assuming that each classroom should receive 30 books. Additionally, if a perfect division is impossible, calculate how many books would remain unassigned. This information will help the school plan any additional purchases or necessary storage space for unused books.

Problem Summary:
The school needs to distribute books such that each classroom receives exactly 30 books. With a total of 720 books available, create a method to determine how many complete classrooms can be supplied and how many books would remain unassigned if an exact division is not possible, all without using division or modulo operators.

Test Cases

Input (Total Books)	Output (complete classrooms books undistributed)
720	24 0
745	24 25
150	5 0
98	6 8
-150	Error! Input must be positive

The output should contain the total number of fully assigned courses on the first line and the number of remaining books on the second line. For negative input, an error message must be displayed

Support Restrictions: If needed, request assistance from the teaching assistant or professor. You have only three attempts for each problem.

Problem 2: Pikachu's Thunder Shock Charge

Pikachu is preparing for a big battle and needs your help managing his electricity levels to ensure he can unleash the strongest Thunder Shock possible. Pikachu charges his electrical energy in one battery per day of the week, distributed in an array of 7 positions. To be well-prepared, Pikachu needs to identify the day with the maximum electricity available in the stored batteries to perform his Thunder Shock.

Your task is to write a program that:
1. Read an array of integers where each integer represents the electrical energy stored in a battery for each day of the week.
2. Identify the position and value of the maximum energy in the array.
3. Display the maximum energy and the corresponding day of the week in words, adding the suffix "is a good day for a Thunder Strike."

Restrictions:
- The array may contain both positive and negative numbers. Negative numbers represent days of energy loss compared to the previous week.
- Days start on Monday (0 = Monday, 1 = Tuesday, ..., 6 = Sunday).

Input:
An array of 7 integers representing the energy in each battery daily.

Output:
A single integer representing the maximum energy Pikachu can gather, followed by the corresponding day in words with the suffix "is a good day for a Thunder Strike."

Test Cases

Entrada	Salida
5 -3 7 2 -8 10 0	10 Saturday is a good day for a Thunder Strike
15 -3 7 2 -8 10 -1	15 Monday is a good day for a Thunder Strike
1 1 1 1 1 1 1	1 Any day is a good day for a Thunder Strike

-2 -4 -5 -1 -1 -1 -10	It's never a good day for a Thunder Strike

Support Restrictions:

If needed, request assistance from the teaching assistant or professor. You have only three attempts for each problem.

Annex 1: ADCP template with example (in case you need it)

Problem: *Sum that adds up*
Build a program that calculates the product of 2 non-negative numbers using only sums.

	Inputs	n,m
ANALYSIS	**Process**	The product of 2 numbers (n, m) can be calculated as the sum n – th of m. $$n * m = m + m + m \dots \text{ n times}$$
	Outputs	Product of m*n
	Constraints	n> 0; m>0

DESIGN

Test cases

# case	Input	Outputs
1	n=5; m=4;	Result=20
2	n=6; m=5;	Result=30
3	n=6; m=0;	Error, should be >0
4	n=-1; m=5;	Error, should be >0

PROGRAMMING

```
1   Proceso suma_que_te_suma
2
3       definir n,m, suma, iterador Como Entero;
4
5       Repetir
6           leer n;
7           si (n<=0) entonces
8               escribir "Error, debe ser positivo";
9           FinSi
10      Hasta Que n>0
11
12      Repetir
13          leer m;
14          si (m<=0) entonces
15              escribir "Error, debe ser positivo";
16          FinSi
17      Hasta Que m>0;
18
19      suma<-0;
20
21      Para iterador<-1 Hasta n Hacer
22          suma<-suma+m;
23      FinPara
24
25
26      Escribir "Resultado= ", suma;
27  FinAlgoritmo
```

TESTING

# case	Test status	Comments
1	☑	
2	☑	
3	☑	
4	☑	