TIC1002: Introduction to Computing and Programming II
Semester II, 2018/2019

Practical Exam 1 (23rd February, 2019)

Task 1. No dinner for Polly? [8 marks]

Polly's teacher asked her to calculate the final value of S-polynomials (Simplified polynomial). Can you help her?

A S-polynomial takes the form of:

$$C_1 * X^1 + C_2 * X^2 + \cdots + C_{n-1} * X^{n-1} + C_n * X^n$$

where C_i are **integer coefficients**. Given a set of coefficients $\mathbf{C_i}$ and the value of **X**, we can calculate the finalvalue (i.e. total) easily. For example, the S-polynomial below:

3X + 2X² + 1X³ can be evaluated to **22** if X is 2. [22 = 3*2 + 2*2² + 1*2³].

3X + 2X² + 1X³ can be evaluated to **54** if X is 3. [54 = 3*3 + 2*3² + 1*3³].

Implement a function where

`int` ***`polynomial`***`(int xValue, int termArray[], int nTerm)`
`xValue` is the value of the X variable in the S-polynomial. `termArray[]` is an array with `nTerm` number of integer coefficients. Note that termArray[0] is C_1, termArray[1] is C_2 ….. termArray[n-1] is C_n.
This function returns the **value of the S-polynomial**.

Additional Requirements

Restrictions (must be respected, otherwise 0 mark)
No predefined math library functions allowed. Your code must make all calculation directly in the function, i.e. no helper function allowed (or needed).

Implementation	Maximum Mark
Iterative (Nested loop of any kind)	**3 marks**
Iterative (Only a single loop)	**5 marks**
Recursive: Without ANY form of loop	**8 marks**

Sample results

`xValue`	`termArray[]`	`nTerm`	Meaning	Return result
2	{3, 2, 1}	3	**$3X + 2X^2 + 1X^3$**	22
3	{3, 2, 1}	3	**$3X + 2X^2 + 1X^3$**	54
11	{7, 5, 3, 2}	4	**$7X + 5X^2 + 3X^3 + 2X^4$**	33957
13	{7, 5, 3, 2}	4	**$7X + 5X^2 + 3X^3 + 2X^4$**	64649

Notes

The given **`printPolynomial()`** function in the template print out the S-polynomial in a more human friendly format on screen.

Task 2. The Land Lord with Unique Taste [12 marks**]

The land lord Mr.Zaleem has a very odd taste. Although he very much like to buy as many land plot as possible, he **couldn't stand to have land plot with the same area size**, i.e. he wants all his land plot to have **unique area size**. As a young land agent, you are tasked to come up with a **list of suitable land plot for Mr.Zaleem.**

Basic Setup

Each land plot is defined by **length** and **width**, as shown as a C++ structure below:

```
struct LandPlot {
    int length, width;    //Area size is calculated from these fields
};
```

Implement two functions

`void sortLandPlot( LandPlot a[], int n )` - **4 marks**
`a[]` is an array with **`n`** number of **`LandPlot`** structures.
This function sorts the land plots in **ascending order** according to their area. **The relative order of land plots with same area size must be retained.**
Restriction: You must use one of the sorting algorithm covered in this course (selection, bubble, insertion). Sample implementation of the sorting algorithms can be found commented at the bottom of the template file.

`int uniqueLandPlot( LandPlot a[], int n )` - **6 marks**
`a[]` is an array with **`n`** number of **`LandPlot`** structures, **sorted in ascending order according to area**.
This function pick the land plots so that the **land plot area size is unique** (i.e. no two land plot has the same area). Whenever there are multiple land plot of the same area size, this function **pick the last one in order (see sample output if you are not sure).** This function returns the number of unique land plots, K. Array a[0…K-1] contains the **unique land plots**, **sorted in ascending order according to area.**
Implementation Approaches: - **[Additional Array] = at most 3 marks** - **[Additional Sorting] = at most 3 marks** i.e. Only solutions with **no extra array** and **no additional sorting** has the potential to get full marks.

Modularity is part of the marking criteria for this task. Use appropriate helper functions.

Sample Output

Below is the sample output from a fully implemented task 2. Note that the printing function is given in the template file to aid your debugging.

Output	Comments
** Original ** (10 x 2: 20) (8 x 2: 16) (1 x 3: 3) (4 x 5: 20) (4 x 4: 16) (1 x 16: 16) (16 x 1: 16) (1 x 1: 1) ** After Sorting ** (1 x 1: 1) (1 x 3: 3) (8 x 2: 16) (4 x 4: 16) (1 x 16: 16) (16 x 1: 16) (10 x 2: 20) (4 x 5: 20) ** After Uniqueness Filter ** 4 unique land plot(s). (1 x 1: 1) (1 x 3: 3) (16 x 1: 16) (4 x 5: 20)	8 Land Plots in the initial array. Format (Length x Width: Area) --- The land plot are in sorted order. Pay attention to the order of these land plots. The relative order from the initial array is **maintained**. (e.g. 8x2 before 4x4) Similarly, 10x2 is before 4x5 in the original. --- The return result of uniqueLandPlot() Note that the **last** land plot is chosen among those with same size. i.e. 16x1 is the last in those with area 16, 4x5 is the last with area 20, etc.

Rewarding Good Programming Style [2 marks]

2 marks out of the 12 marks are used for **programming style** check as follows:

Programming Style	Applicable to	Total
Consistent Indentation	Tasks 1 and 2	1 mark
Good variable naming	Tasks 1 and 2	1 mark

~~~ End of PE Question ~~~