Assignment #4

on C, Pascal, Fortran, PHP, Perl, Cobol, C++, and optional Ada and C#

Date Due: December 8, 2025

Total: 100 marks

Use the specific constructions and the characteristics of each programming language to write the following programs: the input is from standard input, and all output is standard output. Use the constructions from the slides/examples, avoiding built-in functions that are not necessary. You can program these functions yourself (mainly for C++/Php/Perl).

1. (10 marks each program, maximum 80) Write a program in C, also in C++, Pascal, Cobol, Fortran, Perl, PHP, Ada , and C# that computes the sublist S of a list $L = (X_1 \dots X_n)$ obtained as follows:

   (a) The initial list is read from standard input.

   (b) The first element of the sublist S is an even number in list L.

   (c) No even number in L has an index lower than the index of the element selected at the previous step (1b).

   (d) The last element of the sublist S is a number read from standard input and it should be in the list L.

   (e) No number in L has an index higher than the index of the element selected at step 1d.

   (f) All the other elements of the sublist are the squares of the elements of the first list (L), having indexes between the two selected numbers.

   (g) If such numbers (like the ones mentioned at Step 1b or Step 1d) do not exist for the input list, the program should display an appropriate message, and the sublist S is an empty list.

   All the situations generating an empty list should be detected by your program.

   The lists must be stored as arrays.

   You choose the exact format of the input data, but it must be documented (description is in the Readme file).

   Here is a possible example of running a program:

```
$cc x.x -o x
$./x<< HERE
13
105 4 6 3 11 24 47 30 36 47 101 103 7
47
HERE
EOF

4 36 9 121 576 2209 900 1296 47
$
$./x<< HERE2
10
125 301 1367 47 101 10001 41 61 83 24
HERE2
EOF

The list is empty (no even number before other  numbers).
$
$./x<< HERE3
11
125 128 301 1367 47 101 10001 4 6 8 24
6
HERE3
EOF

4 6
$
$./x<< HERE4
11
25 128 301 1367 47 101 10001 4 6 8 24
101
HERE4
EOF

The list is empty (no even number before selected  number).

$
$./x<< HERE5
1
24
24
HERE5
EOF

24
```

2. (10 marks each program, maximum 70 marks) Write the following program in C, and also in C++, Pascal, Cobol, Fortran, Perl, PHP, Ada, and C#:

   Write an interactive program that plays the following game:

   (a) We have stored internally four (4) arrays of size at most M.
   A selection function will select one array of size at most n from the existing four(4) arrays.

n should be read from standard input and $n < M$ must be verified. If $n \geq M$, the user should repeat inputting n. The selection function is created by each of you and arrays cannot be empty. The section function must be different for each of you, so try to be creative. For example, if the arrays are

```
1 4 6 8 9 55
7 100 37 42 96 51 54
8 7 3 5 0 21
8 7 5 900 1200 3400 333
```

and $n = 7$, then you could select an array of length 7: `100 900 333 1 4 5 21` or `100 900 5 1 4 5 21`, but you cannot select `100 900 5 5 4 5 21`, as there are only two 5th in the original arrays (so each element is selected at most once).

(b) Write a program to interactively guess an element from the selected list in at most k tries, where k is also an input (integer) value $n < k < 3n$. If k is out of bounds, the user should reenter the value of k. The program should precalculate the maximum and minimum number from all lists, and should tell the user if the number is out of bounds. Therefore, your input k is followed by at most k tries.

(c) The program will determine if the numbers are in the arrays. If yes, then it is a successful guess, i.e., the number guessed, say r, and the program will reveal all the numbers in the array.

In other words, once you guess a number in one array, you display all the numbers in that array and stop.

We don't know how the numbers are distributed between the arrays. Certain values might be present in a few arrays, or in all arrays.

After each guess, successful or not, the number of tries left should be displayed.

The player wins if all numbers are revealed after at most k tries and loses otherwise.

Do not copy any code from the internet. Create your own selection functions!

10 marks (each program)

3. **Additional program:** (10 marks each program); the maximum of the assignment can be increased by 10 marks if you solve this problem in any of the listed languages. It could also compensate for the other two problems that are not implemented (in any languages).

Examples:

(a) If Problem 1 is not implemented in C, you can solve this problem in C, and you can get the missing points for Problem 1.

(b) If for Problem 1 you can get 67 marks, it means that some programs are marked lower. The marks for solving Problem 3 can replace some of the lower marks for Problem 1. You cannot use Problem 3 to replace two programs (one for Problem 1, and one for Problem 2 – it will replace only the lower marked program).

(c) If you write all Programs for Problem 1 (80 marks) and Problem 2 (70 marks), and you write two programs for Problem 3, you may get 160 (as this is the upperbound) – one program is considered a replacement and the second one bonus.

The following program can be written in any of the languages C, C++, Pascal, Fortran, Cobol, Perl, PHP, C#, and Ada.

Write a function which computes: $\sum_{i=0}^{n}(f(3i^2-1)^3 - f(i-1)^3)$, where f is a polynomial function of degree at least 3 (the choice of f is yours).

Advice to get the maximum marks: Write the code in one language, then "translate it" to the other ones. You may have the same algorithm in every language, but at the same time, you should take advantage of the characteristics/structure/data types/built-in types of that language. If you just have a blind translation without taking into consideration that it is a different language, your mark will be lower. For example:

1. You don't use a function call (like in java) to modify the third element of an array in C, you modify it directly.

2. You cannot have the same code for C and C++. Moreover, you need to have C++ characteristic constructions.

3. Some programming languages allow classes and recursion, others don't. Adapt your algorithm accordingly.

Marking scheme: 10 marks (each), including program testing (the `run.txt` file). (maximum 150/160 – if solving problem 3).

Important: Please write your own code, don't share your code, don't use "foreign code".
If you plagiarize, the mark is automatically 0 for the **entire course**, regardless of any other marks.