FUTUREGAMES

Basic programming challenges

- In C++

Krister Cederlund

krister.cederlund@futuregames.nu

Futuregames

CONTENTS

INFORMATION

In this booklet you will find several challenges testing you on different programming concepts and skills, some of which might be common in job interviews in some form.

Many of the problems are language agnostic but will have a C++ focus. If any of the exercises requires some boilerplate code, you will be able to find them in your Course.

SOLVING PROBLEMS

The one technique you should try to avoid when solving problems is trial and errors. Rather analyze the problem so you fully understand what it is you are trying to solve. What is the outcome?

Step away from the computer and try solving it on paper first and when you are confident you understand the problem, try and solving it with code. If you were wrong in your thinking or you did not understand the problem, go back to paper, and try again.

Discussing the problem with others is also a good way of trying to figure the problem out.

When you think you have solved the problem in code, test with multiple outcomes, does it give the same result you have on paper?

Try to solve the problem in multiple ways in code, what are the pros or cons of each solution, which one is easier to read, more performant, ask yourself question regarding your code and try to be critical of it.

Compare your solution(s) to others and ask similar questions comparing them.

When trying to solve larger problems, try breaking them down into sub-problems or sub-programs, each sub-problem should be able to be solved independently of the others and then combined into a complete solution for the larger problem.

ASKING FOR HELP

You should never be afraid to ask for help, but you need to have done the leg work first and show it.

First, describe the overall program you're working in, summarize it to give them a hint of what you are working on.

EXAMPLE

> I'm making a third person camera in Unity.

Then go into details, what are your actual problem, be detailed. Explain the expected outcome and your actual outcome.

EXAMPLE

> I'm trying to get the camera to orbit around the player by moving the mouse. When I move the mouse to the right it should orbit around the player to the right side as well. Instead, nothing happens.

Explain the tries you have made so far, show the one trying to help you that you have tried to solve the problem before asking for help and that you just didn't give up and want someone else to do the work for you.

Summarize the different solutions you have tried, what outcome you expected and what outcome you got.

EXAMPLE

> I have looked that my method is executed, that everything is getting called as it should using Debug.Log to make sure, and it does.
> I have also logged the values from the input expecting to see the values of the delta movement of the mouse, but I get zeroes instead and I don't know why.

Then you can start working with the other people trying to solve the problem together. When being helped, always make sure you are active in the process. The one helping should not solve the problem for you, but rather together help you find the solution or come up with viable ideas that can lead you to new ideas trying to solve the problem at hand.

WEEK 01

ABSOLUTE DIFFERENCE - EASY

Given two numbers, implement a method returning the absolute difference between the numbers.

CONSTRAINTS

- No constraints.

SAMPLE INPUT

```
Number A = -6, Number B = 4
```

SAMPLE OUTPUT

```
The absolute difference is 10
```

INCREASE DIFFICULTY

- Implement the Abs method yourself without using a ready-made abs function.

CELCIUS TO FAHRENHEIT - EASY

Given a decimal value representing degrees in Celsius, convert the value to Fahrenheit.

What result do you get when you use integers instead, and why?

Formula to convert Celsius to Fahrenheit: Celsius * 9 / 5 + 32.

CONSTRAINTS

- No constraints.

SAMPLE INPUT

```
Celsius = 36
```

SAMPLE OUTPUT

```
36 degrees Celsius is 96.8 degrees Fahrenheit
```

EVEN OR ODD - EASY

Given an integer value, implement a method that returns if it is an even number or an odd number.

CONSTRAINTS

- No constraints.

SAMPLE INPUT

```
Number = 3
```

SAMPLE INPUT

```
3 is an odd number
```

LAST DIGIT OF A NUMBER - EASY

Given an integer value, implement a method that returns the last digit of the number

CONSTRAINTS

- No constraints.

SAMPLE INPUT

```
Number = 137
```

SAMPLE INPUT

```
7
```

INCREASE DIFFICULTY

- Print each digit separately (1 3 7).
- Print each digit separately in reverse order (7, 3, 1).

SWAP - EASY

Create a program that inputs two numbers, create three ways to swap its values, one that swaps by using STL, one that swaps by reference and one that swaps by pointers.

CONSTRAINTS

- No constraints.

SAMPLE INPUT

```
Number A = 3, Number B = 7
```

SAMPLE INPUT

```
Swapping
Number 7 = 3, Number B = 3
```

INCREASE DIFFICULTY

- Perform the swap using only mathematics without a temporary variable.

GUESSING GAME - EASY

Write a program that has a secret number. Let the user guess the number until they guess the correct number. Then present the number of tries it took to for the user to guess the correct number.

CONSTRAINTS

- No constraints.

SAMPLE INPUT

```
Guessed numbed = 10
```

SAMPLE OUTPUT

```
10 is correct, it took you 7 tries.
```

INCREASE DIFFICULTY

- Randomize the secret number between a minimum and maximum number.
- Handle faulty input.
- When the player has played one session, ask them to play again or not without having to restart the program.

ARITHMETIC PROGRESSION - EASY

Arithmetic progression is when the difference between each successive pair of values in a series of numbers are the same.

Given the values: 2, 4, 6, 8 the difference between is always two meaning it have arithmetic progression.

Write a program with a function that returns whether a series of numbers have arithmetic progression or not.

CONSTRAINTS

- No constraints.

SAMPLE INPUT

```
Numbers = 2, 4, 6, 8
```

SAMPLE OUTPUT

```
Series has arithmetic progression.
```

REVERSE COLLECTION - EASY

Write a program with a collection and reverse the collection before presenting it to the user.

CONSTRAINTS

- Do not use any STL functionality, you may use a string as the collection.

SAMPLE INPUT

```
String = Hello
```

SAMPLE OUTPUT

```
olleH
```

LINEAR SEARCH - EASY

Given a collection of integers, implement a method to search for a certain integer and return the position in the collection where it was found, or a negative value if it was not found.

CONSTRAINTS

- Do not use any built-in or STL functionality for finding the value.

SAMPLE INPUT

```
Collection { 5, 3, 7, 10, 1 }, Search for = 3
```

SAMPLE INPUT

```
The number 3 was located at position 1
```

INCREASE DIFFICULTY

- Return all positions where the number looked for was found.

PALINDROME - EASY

A palindrome is a word that is spelled the same way backwards, like “eye”.

Implement a method with a string parameter and evaluates whether the word is a palindrome or not.

CONSTRAINTS

- Do not use any STL functionality.

SAMPLE INPUT

```
Word = eye
```

SAMPLE OUTPUT

```
The word “eye” is a palindrome.
```

INCREASE DIFFICULTY

- Implement the same method using recursion.

NTH BIT OF A NUMBER - EASY

Implement two methods, one for retrieving the nth bit of a number and, one for setting the nth bit of a number

CONSTRAINTS

- Use bitwise operators.

SAMPLE INPUT

```
Numbers = 12
```

SAMPLE OUTPUT

```
Set bit 0
Number is now 13

Get bit 2

Bit 2 is set to 1
```

SWAP BITWISE - EASY

Create a program that inputs two numbers, create a function to swap its values using only bitwise operators.

CONSTRAINTS

- Use only bitwise operators.

SAMPLE INPUT

```
Number A = 3, Number B = 7
```

SAMPLE INPUT

```
Swapping
Number 7 = 3, Number B = 3
```

INCREASE DIFFICULTY

- Implement a method to perform the swap but print the result outside of the method.
- Perform the swap using only mathematics.

MIN MAX SUM AVERAGE - EASY

Given an array of integers, write a program that presents the sum, the lowest value, the highest value, and the average (mean) of all numbers in the array.

CONSTRAINTS

- Don't use any built-in functionality except for settings initial values.

SAMPLE INPUT

```
Numbers = 9, 5, 2, 8, 3
```

SAMPLE OUTPUT

```
Sum = 27, Minimum number = 2, Highest number = 9, Average = 5.4
```

CONSTRUCTOR ORDER IN INHERITENCE - EASY

Write a base class A and two other classes B and C, B derives from A and C derives from B. In their constructors and destructors output something to the console and reflect about the order the constructors and destructors are called in a chain of inheritance.

CONSTRAINTS

- No constraints.

SAMPLE INPUT

```
No input.
```

SAMPLE OUTPUT

```
Constructor A called
Constructor B called
…
```

OCCURENCES OF CHARACTERS IN A STRING - INTERMEDIATE

Given a line of text, write a program that presents how many occurrences of each character is in the text.

CONSTRAINTS

- Store the occurrences in a collection, you can use array(s) or an std::map.
- Only present the letters that appear in the text.
- You can treat each character as uppercase.

SAMPLE INPUT

```
String = Hello
```

SAMPLE OUTPUT

```
H = 1
E = 1
L = 2
O = 1
```

FIBONACCI NUMBERS - INTERMEDIATE

The Fibonacci numbers form a sequence in which each number is the sum of the two preceding numbers.

Write a program that receives an integer. It should then continue to print the Fibonacci sequence up till and including that number.

Implement it in two ways, by recursion and iteratively.

CONSTRAINTS

- No constraints.

SAMPLE INPUT

```
Number = 9
```

SAMPLE OUTPUT

```
0, 1, 1, 2, 3, 5, 8, 13, 21, 34
```

LUCKY NUMBER - INTERMEDIATE

Given an integer value, write a program to determine if the number is lucky or not.

A number is lucky if all the digits in the number is unique.

CONSTRAINTS

- No constraints.

SAMPLE INPUT

```
Number = 1224
```

SAMPLE OUTPUT

```
The number 1224 is not lucky.
```

LINEAR SEARCH WORD - INTERMEDIATE

Given a string of words, implement a method looking if the sentence contains a specific word.

CONSTRAINTS

- Do not use any built-in functionality.

SAMPLE INPUT

```
Sentence = “Hello my name is Ada”, Look for = “name”.
```

SAMPLE INPUT

```
The sentence contains the word “name”.
```

SWAP - INTERMEDIATE

Given an array of integers, write a program that presents the sum, the lowest value, the highest value, and the average (mean) of all numbers in the array.

CONSTRAINTS

- Use bitwise operators.
- No temporary variable.

SAMPLE INPUT

```
Number A = 3, Number B = 7
```

SAMPLE OUTPUT

```
Swapping
Number 7 = 3, Number B = 3
```

CONTAINS NUMBER - HARD

Given two integers, number A must be a single digit, implement a method that returns if any of the digits of number B contains the digit of number A.

CONSTRAINTS

- No constraints.

SAMPLE INPUT

```
Number A = 4, Number B = 341
```

SAMPLE OUTPUT

```
341 contains the number 4.
```

PERMUTATIONS - HARD

A permutation is a way to find all possible arrangements of a set.

Write a program that prints all possible permutations of a string.

CONSTRAINTS

- No constraints.

SAMPLE INPUT

```
String = ABC
```

SAMPLE OUTPUT

```
ABC, ACB, BAC, BCA, CBA, CAB
```

COMPRESS STRING - HARD

Implement a method that compresses a string by counting the sequence of occurring characters in a string.

Also implement a method to decompress a string.

CONSTRAINTS

- No constraints.

SAMPLE INPUT

```
String = aaaabbbcc
```

SAMPLE OUTPUT

```
a4b3c2
```

WEEK 02

FIND LARGEST NUMBER - EASY

Implement a template function to compare two values.

CONSTRAINTS

- Use function template.

SAMPLE INPUT

```
ValueA = 3, ValueB = 7
```

SAMPLE OUTPUT

```
ValueB (7) is higher than ValueA (3)
```

SWAP WITH TEMPLATES - EASY

Implement a function template taking to parameters and swapping them.

CONSTRAINTS

- Use templates.

SAMPLE INPUT

```
Number A = 3, Number B = 7
```

SAMPLE INPUT

```
Swapping
Number 7 = 3, Number B = 3
```

INCREASE DIFFICULTY

- Perform the swap using only mathematics without a temporary variable.

CUSTOM VECTOR CLASS - HARD

Implement your own container like List in C# and STL::Vector in C++.

The vector class should contain the following functionality.

- push_back(data): Adds an element to the end of the array.
- pop_back(): Removes an element from the end of the array and returns the popped element.
- size(): Returns the current size of the vector.
- Make it iterable (usable in a foreach loop).

CONSTRAINTS

- Use templates for the containers data type.