1/3rd term Programming Fundamentals

Oct. 4 2024, 18:30-21:30h

- You can solve the problems in any order. Solutions must be submitted to the automated judgement system Themis. For each problem, Themis will test ten different inputs, and check whether the outputs are correct.
- Grading: you get one grade point for free. The remaining nine points are based solely on the judgment given by Themis. The first problem is worth one grade point. The remaining four problems are worth two grade points each.
- You can score partial points if a program passes several but not all test cases. For example, if you pass 6 out of 10 test cases, then you are awarded 60% of the grade points for that exercise.
- Inefficient programs may be rejected by Themis. In such cases, the error will be 'time limit exceeded'. The time limit for each problem is one second.
- The number of submissions to Themis is not limited. No points are subtracted for multiple submissions.
- There will be no assessment of programming style. However, accepted solutions are checked manually for cheating: for example, precomputed answers will not be accepted, even though Themis accepts them.
- Needless to say: you are not allowed to work together. If plagiarism is detected, both parties (supplier of the code and the person that sends in copied code) will be excluded from any further participation in the course.
- You are not allowed to use email, phones, tablets, calculators, etc. There is a calculator available on the exam computers (see icon on the desktop). You are allowed to consult the ANSI C book and a dictionary. You are not allowed to use a printed copy of the reader or the lecture slides, however they are available digitally (as a pdf) in Themis. You are allowed to access your own submissions previously made to Themis.
- For each problem, the first three test cases (input files) are available on Themis. These input files, and the corresponding output files, are called `1.in` , `2.in`, `3.in`, `1.out`, `2.out` and `3.out`. These files can be used to test whether the output of your program matches the requested layout, so that there can be no misunderstanding about the layout and spaces in the output.
- **If you fail to pass a problem for a specific test case, then you are advised not to lose much time on debugging your program, and continue with another problem. In the last hour of the exam, all test cases will be made visible in Themis.**

Problem 1 is worth 1 grade point. The other exercises are worth 2 grade points (totaling 9 points, you get 1 point for free).

Problem 1: Powerful

Consider the decimal number 78346. Its least significant digit is 6 and has position 0, while its most significant digit 7 has position 4. We take the sum of its digits raised to the power of their position. In this case we get $7^4 + 8^3 + 3^2 + 4^1 + 6^0 = 2401 + 512 + 9 + 4 + 1 = 2927$.

Write a program that reads from the input a non-negative `int`. The output should be the number that is obtained by summing its digits raised to their positions. Note that $0^0 = 1$ by definition.

Example 1:

input:
```
78346
```
output:
```
2927
```

Example 2:

input:
```
42
```
output:
```
5
```

Example 3:

input:
```
123456789
```
output:
```
2781
```

Problem 2: Animal Day

World Animal Day is an international day of action for animal rights and welfare celebrated annually on October 4, the feast day of Francis of Assisi, the patron saint of animals.

Today it is animal day because the date is Friday the 4th of October 2024. The next year in which the 4th of October is again a Friday is 2030.

Recall that a year has 365 days, except for *leap years* which have 366 days. A year is a leap year if it is divisible by 4, but not by 100. Exceptions are years that are divisible by 400 (these are leap years).

Write a program that reads from the input a positive integer `n` (where $n < 1000$), and outputs the `n`th next year (starting from 2024 for `n=0`) in which animal day is a Friday.

Example 1:

input:
```
0
```
output:
```
2024
```

Example 2:

input:
```
1
```
output:
```
2030
```

Example 3:

input:
```
2
```
output:
```
2041
```

Problem 3: Primal Sum Iterations

Let n be an integer greater than 1. We apply the following process.

1. If $n = 10$ or $n = 30$ or n is a prime number then the process stops.
2. Otherwise, let s be the sum of the distinct prime factors of n.
3. Let m be the multiplication of s with the number of distinct prime factors of n.
4. Replace n by m and go to step 1.

It can be shown that this process terminates for any integer $n > 1$. The proof is not very hard, but you may simply assume this claim to be true.

As an example, consider $n = 93$. The steps of the algorithm are given in the following table.

n	factorization of n	s	m
93	3×31	$3 + 31 = 34$	$34 \times 2 = 68$
68	$2 \times 2 \times 17$	$2 + 17 = 19$	$19 \times 2 = 38$
38	2×19	$2 + 19 = 21$	$21 \times 2 = 42$
42	$2 \times 3 \times 7$	$2 + 3 + 7 = 12$	$12 \times 3 = 36$
36	$2 \times 2 \times 3 \times 3$	$2 + 3 = 5$	$5 \times 2 = \mathbf{10}$

We conclude that it takes 5 iterations to reach 10 from 93 ($93 \rightarrow 68 \rightarrow 38 \rightarrow 42 \rightarrow 36 \rightarrow 10$). Clearly, the number of iterations needed depends on the starting number.

Write a program that reads from the input two integers a and b (where $2 \leq a \leq 10^6$ and $a \leq b \leq 10^6$), and outputs the number n (where $a \leq n \leq b$) for which the above process makes the maximum number of iterations followed by the number of iterations. If there exist multiple numbers in the interval that reach this maximum number of iterations then the smallest number needs to printed.

Example 1:
input:
```
42 93
```
output:
```
93 5
```

Example 2:
input:
```
2 100
```
output:
```
93 5
```

Example 3:
input:
```
997 997
```
output:
```
997 0
```

Problem 4: Impact

Consider the following sequence of integers: `3, 4, 1, 2, 5, 3, 1, 5, 3, 4`.
The *impact* of a number n is defined as n multiplied with the number of times it occurs in the sequence. So, the impact of the number 3 is $3 \times 3 = 9$, and the impact of 5 is 5×2.

The input of this problem is a non-empty sequence of positive integers terminated by the number 0. The numbers in the sequence are at most 1000 You may assume that the impact of each number fits in a stanfdard `int`. The output must be the number from the sequence with the highest impact. If multiple numbers have this same highest impact, then the smallest of these numbers must be printed.

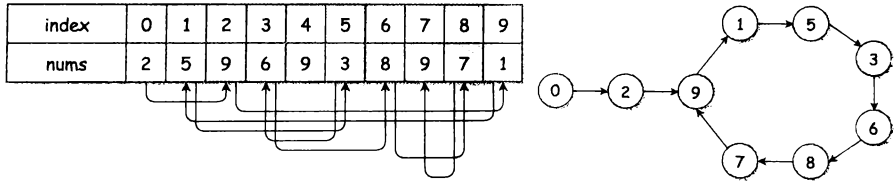
Example 1:
input:
```
3 4 1 2 5 3 1 5 3 4 0
```
output:
```
5
```

Example 2:
input:
```
1 1 1 3 2 0
```
output:
```
1
```

Example 3:
input:
```
1 1 1 4 2 0
```
output:
```
4
```

Problem 5: Cycle Detection

The input for this problem is an integer array `nums[]` with length `n` (where $1 \leq n \leq 100000$). For every index `i` (with $0 \leq$ `i` $< n$) it is given that $0 \leq$ `nums[i]` $<$ `n`. As an example, consider the array `nums=[2,5,9,6,9,3,8,9,7,1]` (i.e. `n=10`).



We traverse through the array from the starting index `idx=0`. Next, we inspect `nums[idx]` and update `idx` to that value. This traversing is repeated until we reach a value that we have reached before That value is the *entrance point* of a cycle, and should be printed on the output together with the length of the cycle. So, for the given example, the output would be "`9 7`" because 9 is the entrance point of a cycle that consists of 7 steps. This cycle is detected after successively visiting the indexes $0 \to 2 \to 9 \to 1 \to 5 \to 3 \to 6 \to 8 \to 7 \to 9$.
The input for this problem is a line containing the number `n` ($1 \leq n \leq 100000$, the length of the array), followed by a line with the content of the array. The output should be as described above.

Example 1:
input:
```
10
2 5 9 6 9 3 8 9 7 1
```
output:
```
9 7
```

Example 2:
input:
```
5
1 2 3 4 2
```
output:
```
2 3
```

Example 3:
input:
```
5
1 2 3 4 4
```
output:
```
4 1
```