Resit 1/3rd term Programming Fundamentals

Oct. 13 2023, 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: Epoch

Standard Unix systems have a built-in function `time()` that returns the time as the number of seconds since the *Epoch*. The Epoch is defined as 1970-01-01 00:00:00 (1 January 1970, 00:00:00). This day was a Thursday.

Write a program that reads from the input a positive integer n, which is the number of seconds since the epoch. The output should be the day of the moment in time that is n seconds after the epoch. As an example, For $0 \leq n < 24 * 3600 = 86400$ the output should be `Thursday`, but for $n = 24 * 3600 = 86400$ the output should be `Friday`.
To avoid confusion about the spelling of the days of the week, the output should be any of [`Monday`, `Tuesday`, `Wednesday`, `Thursday`, `Friday`, `Saturday`, `Sunday`].

Example 1:
input:
```
0
```
output:
```
Thursday
```

Example 2:
input:
```
86399
```
output:
```
Thursday
```

Example 3:
input:
```
86400
```
output:
```
Friday
```

Problem 2: Friday the 13th

Today, it is Friday the 13th of October 2023. The next year in which the 13th of October is again a Friday is 2028. 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 $\mathtt{n} < 1000$), and outputs the `n`th next year (starting from 2023 for `n=0`) in which the 13th of October is a Friday.

Example 1:
input:
```
0
```
output:
```
2023
```

Example 2:
input:
```
1
```
output:
```
2028
```

Example 3:
input:
```
2
```
output:
```
2034
```

Problem 3: Circular primes

An *circular prime* is a prime number which remains prime on any cyclic rotation of its digits. For example, 1931 is a prime number, and so are 9311, 3119, and 1193. Hence, 1931, 9311, 3119, and 1193 are circular primes. The number 997 is prime, but it is not a circular prime because 979 is not prime ($979 = 11 \times 89$).

The input for this problem is an integer n, where $0 \leq n \leq 10^6$. The output must be `YES` if n is a circular prime, and `NO` otherwise.

Example 1:	**Example 2:**	**Example 3:**
input:	**input**:	**input**:
`1931`	`997`	`719`
output:	**output**:	**output**:
`YES`	`NO`	`YES`

Problem 4: Isograms

An *isogram* is a word with no repeating letters. For example, `palindrome` is an isogram, while `palindromic` is not because it contains the letter `i` twice.

The input is a sentence having at most 200 characters, terminated by a period (`'.'`). The output must be all isograms (in lower case) from the sentence in the same order as they appear in the input, separated by a single space character. The output must be terminated by a newline (`\n`) character, even in the case that there are no isograms in the sentence. Note that any non-alphabetic input character is not part of a word and should be skipped (including newlines).

Example 1:

input:
```
Former president Trump is against a republican strategy that will
allow fairer elections.
```
output:
```
trump is a republican
```
Example 2:

input:
```
You should prevent ChatGPT hallucinating programs because testing
these will kill design skills that your fellow students developed
designing algorithms.
```
output:
```
you should design your algorithms
```
Example 3:

input:
```
Logicians state that they will accept that 1+1=3 according
axiomatic identities that say that it is true that 1=2.
```
output:
```
they say it is true
```

Problem 5: Most frequent value

The input of this problem consists of several lines. Each line contains two integers `a` and `b`, where $0 \leq$ `a` $\leq$ `b` ≤ 10000. The last line of the input is the pair `a=b=0`. Each pair `a`, and `b` represents a consecutive half-open interval `[a,b)`. For example, $[0,5) = [0,1,2,3,4])$.

The output of the program must be the smallest value `n` that occurs most frequent in the input intervals.

Example 1:

input:
```
40 100
42 50
0 0
```
output:
```
42
```

Example 2:

input:
```
0 10
5 6
6 8
2 9
0 0
```
output:
```
5
```

Example 3:

input:
```
8393 9952
6810 7703
2276 3597
8446 8849
5260 5539
0 0
```
output:
```
8446
```