2018 年度 情報領域演習第三
— 第 2 回 —

はじめに

前回同様，CED において以下のコマンド (青字の部分) を実行することで出席を表明できる．ただし，N は所属するクラス名 1 ，2 ，3 で置き換え，N の前には空白を入れないこと．

```
[p1610999@blue00 ~]> /ced-home/staff/18jr3/02/checkerN
提出開始: 10 月 xx 日 14 時 40 分
提出締切: 10 月 yy 日 14 時 39 分
ユーザ: p1610999, 出席状況: 2018-10-05-14-58
 問題:    結果 | 提出日時           | ハッシュ値                        |
    1:  未提出 |                    |                                  |
    2:  未提出 |                    |                                  |
                        ⋮
```

出席を表明するには，授業の開始から 30 分以内に実行する必要がある．実行しても出席状況が「欠席」である場合は教員に伝えること．なお，上の出力は例であるので，実際の締切りは各自コマンドで確認せよ．

提出も同じコマンドを用いて以下のように実行する．ただし，N はクラス名 (1 から 3)，X は 1 から 7 のいずれかの問題番号であり，prog.c は提出する C プログラムのファイル名であり，X の前後には忘れずに空白を入れること．

```
[p1610999@blue00 ~]> /ced-home/staff/18jr3/02/checkerN X prog.c
```

ここで，ファイル名として指定するのは，問題番号 X の問題を解く C プログラムのソースファイルであり，コンパイル済みの実行ファイルではない．ファイル名は特に指定しないが，「英数字からなる文字列.c 」などとすることが望ましい．このコマンドを実行することにより，指定されたファイルがコンパイルされ，実行テストプログラムが自動的に起動される．コンパイルに失敗した場合にはエラーとなり，提出されたことにはならないので注意すること．コンパイルが成功した場合には複数回の実行テストが行われ，実行テストにも成功すると「成功」と出力される．実行テストに失敗すると「失敗」と出力されるとともに反例が出力されるので，失敗した理由を見つけるための参考にするとよい．ただし，入力のない問題の場合は失敗しても反例は出力されない．

正しく提出できたか確認するためには以下のように出席の表明と同じコマンドを用いて確認できる．

```
[p1610999@blue00 ~]> /ced-home/staff/18jr3/02/checkerN
提出開始: 10 月 xx 日 14 時 40 分
提出締切: 10 月 yy 日 14 時 39 分
ユーザ: p1610999, 出席状況: 2018-10-05-14-47
 問題:    結果 | 提出日時           | ハッシュ値                        |
    1:  成功 | 2018-10-05-15-33   | 9b762c81932f3980cf03a768e044e65b |
                        ⋮
```

ハッシュ値は，提出した C プログラムのソースファイルの MD5 値である．CED では md5sum コマンドを用いて MD5 値を見ることにより，提出したファイルと手元のファイルが同じものであるかを確認することができる．

```
[p1610999@blue00 ~]> md5sum prog.c
9b762c81932f3980cf03a768e044e65b
```

なお，一度「成功」となった問題に対し，実行テストに失敗するプログラムを送信してしまうと，結果が「失敗」になってしまうので注意すること．

問題 1

標準入力から与えられた整数 n に対し，フィボナッチ数列の n 番めを 2018 で割った余りを標準出力に出力するプログラムを作成せよ．ただし，フィボナッチ数列とは，$F_0 = 0$，$F_1 = 1$，$F_n = F_{n-1} + F_{n-2}$ $(n \geq 2)$ で定義される数列であり，入力として与えられる文字列は `int` 型の 10 万以下の非負整数を表すものとする．

入力例

```
21
```

出力例

```
856
```

入力例

```
31415
```

出力例

```
1223
```

ヒント

- $n + m$ を d で割った余りは,「n を d で割った余り」と「m を d で割った余り」を足したものを d で割った余りに等しい．
- 単純な再帰にすると何度も同じ計算が行われるために，時間がかかってしまうことに注意せよ．最初に配列を用意し，添字の小さい方から順番に，i 番めに「F_i を 2018 で割った余り」を格納していくとよい．たとえば，配列を用いて F_0 から F_{49} までを効率よく計算するには以下のようにすればよい．

```
1 int i, fib[50] = {0,1};
2 for(i=2;i<50;++i) fib[i] = fib[i-1] + fib[i-2];
```

問題 2

標準入力から与えられた 2 つの非負整数 n と k に対し，組合せの数 ${}_n\mathrm{C}_k$ を 2018 で割った余りを標準出力に出力するプログラムを作成せよ．ただし，${}_n\mathrm{C}_k$ は ${}_n\mathrm{C}_k = \dfrac{n!}{k!(n-k)!}$ によって計算される数であり，入力として与えられる文字列は 2 つの `int` 型の非負整数 n と k が空白で区切られた文字列であり，$k \leq n \leq 1000$ であるものと仮定してよい．

入力例

```
5 2
```

出力例

```
10
```

入力例

```
500 200
```

出力例

```
1910
```

ヒント

- ${}_n\mathrm{C}_k$ $(0 \leq k \leq n)$ は以下のように再帰的に計算できることを利用するとよい．

$$
{}_n\mathrm{C}_k = \begin{cases} 1 & k = 0 \text{ または } k = n \\ {}_{n-1}\mathrm{C}_k + {}_{n-1}\mathrm{C}_{k-1} & 0 < k < n \end{cases}
$$

  つまり，$0 < k < n$ のときには，${}_n\mathrm{C}_k$ を 2018 で割った余りは,「${}_{n-1}\mathrm{C}_k$ を 2018 で割った余り」と「${}_{n-1}\mathrm{C}_{k-1}$ を 2018 で割った余り」を足して 2018 で割った余りと等しくなる．

- 前問と同様に単純な再帰にすると何度も同じ計算が行われるために，時間がかかってしまうことに注意せよ．2 次元配列を用意し，添字の小さい方から順番に，i 番めの配列の j 番めに「${}_i\mathrm{C}_j$ を 2018 で割った余り」を格納していくとよい．

問題 3

標準入力から順に与えられた 2 つの整数 $a > 0, b \geq 0$ に対し，a^b を 2018 で割った余りを標準出力に出力するプログラムを作成せよ．ただし，入力として与えられる文字列は `int` 型の正の整数と非負整数が空白で区切られた文字列であり，例えば 3^{100} なら `3 100` のように与えられる．

入力例

```
3 100
```

出力例

```
1255
```

入力例

```
12345 67890
```

出力例

```
787
```

ヒント

- この問題は偶数と奇数の場合に分けて再帰で書くとよい．
- $n^{k+1} = n^k \times n$ なので，n^{k+1} を d で割った余りは，「n^k を d で割った余り」と「n を d で割った余り」を掛けたものを d で割った余りに等しい．
- $n^{2k} = (n^k)^2$ なので，n^{2k} を d で割った余りは，「n^k を d で割った余り」を 2 乗したものを d で割った余りに等しい．

問題 4

標準入力から与えられた文字列に対し，先頭の文字を 1 番め と数えるとき，素数番めの文字のみを出力するプログラムを作成せよ．ただし，空白も文字として数えるものとし，入力は 1000 文字以下と仮定してよい．

入力例

```
abcdefg 12345678
```

出力例

```
bceg35
```

入力例

```
The University of Electro-Communications
```

出力例

```
heUistfEtmui
```

問題 5

標準入力から与えられた複数行の文字列に対応する図形を標準出力に描画せよ．入力の各行には 0 個以上の数字が空白文字で区切られた文字列 $a_1, a_2, \ldots, a_n$ が与えられ，それに対する出力では，同じ行の a_1 番め，a_2 番め，$\ldots$，a_n 番めの文字が ‘`#`’，それ以外が空白となっている．ただし，先頭の文字は 0 番めであり，各行において最後の ‘`#`’ の直後は改行するものとし，数字が何も含まれない行は改行のみを出力する．なお，入力の各行の文字列は 128 文字以下であり，各整数値は 60 以下であると仮定してよい．

入力例

```
0 3 5 6 7 8 11 12 13
0 3 5 10
0 3 5 6 7 8 10
0 3 5 10
1 2 5 6 7 8 11 12 13
```

出力例

```
#  # ####  ###
#  # #    #
#  # #### #
#  # #    #
 ##  ####  ###
```

ヒント

- 標準入力を一行ずつ入力するプログラムの例として，標準入力の各行をそのまま標準出力に出力するプログラムを以下に用意したので参考にするとよい．

```
1 #include <stdio.h>
2 #include <string.h>
```

```
3  #define MAXLEN 128  /* 文字列の最大長 */
4  char buf[MAXLEN+2]; /* 文字列を読み込む配列 (改行文字とNULL 文字を含む) */
5
6  int main() {
7    char *p;
8
9    while( fgets(buf, sizeof(buf), stdin) != NULL ) {
10     p = strchr(buf, '\n');
11     if (p != NULL) *p = '\0';
12     printf("%s\n", buf);
13   }
14   return 0;
15 }
```

ここで，関数 strchr(char *str,char c) は文字列 str に文字 c が存在すればその位置を示すポインタを返し，存在しなければ NULL ポインタを返す関数である．上のプログラムの 10–11 行めでは，改行文字が存在すればその文字をナル文字 (‘\0’) にすることで改行を削除している (この問題については必ずしもこの処理を行う必要はないが，後の問題に役に立つことがあるので覚えておくとよい)．

- 文字列から整数を 1 つずつ取り出すには以下のように sscanf を繰り返し用いるとよい．たとえば，buf に文字列が格納されているとき，1 つずつ数値を読み込んで出力するプログラムは次のように書ける．

```
1    char *p = buf;
2    int i;
3    while ( sscanf(p,"%d",&i) != EOF ) {
4      printf("%d\n",i);
5      p = strchr(p,' ');          /* p を次の空白の場所に移動 */
6      if (p == NULL) break;       /* 空白が見つからなければ終了 */
7      else while (*p == ' ') ++p; /* 見つかったら空白でなくなる位置まで移動 */
8    }
```

- 課題のページに入力サンプルをいくつか用意しているので試してみるとよい．たとえば，サンプルファイル input.txt をダウンロードした場合，コンパイルしたプログラムが a.out であるならば，

```
[p1610999@blue00 ~]> ./a.out < input.txt
```

とすればよい．

問題 6

標準入力から与えられた一行の文字列に対し，文字列の長さを標準出力に出力するプログラムを作成せよ．ただし，スラッシュ記号 ‘/’ が 2 つ並んだ文字列 ‘//’ が現れたときは，(複数あるときは最初の) ‘//’ を含めて文字列の終わりまで長さには数えないものとする．また，改行文字は含めないが，空白文字や他の制御文字も 1 文字と数える．入力される文字列の長さは ‘//’ 以降も含めて 128 文字以下で，ASCII 文字のみで構成されると仮定してよい．

入力例

```
abcd
```

出力例

```
4
```

入力例

```
a b c d/1234//x y z//xyz
```

出力例

```
12
```

問題 7

標準入力から与えられた 2 つの文字列 w_1 と w_2 に対し，w_1 の中に w_2 が含まれているかどうかを判定し，含まれている場合はその位置 (先頭の文字は 0 番め) を含まれていない場合は`-1` を標準出力に出力するプログラムを作成せよ．ただし，w_1 は 1 行めの，w_2 は 2 行めの改行の前までの文字列とし，入力となる文字列は 2 つとも 128 文字以下であると仮定してよい．なお，この問題については `string.h` に含まれる関数を使わないこと [1]．使った場合でも「成功」と表示されるが採点の際には得点にはならないので注意せよ．

入力例

```
The University of Electro-Communications
nica
```

出力例

```
31
```

入力例

```
Practice makes perfect.
esp
```

出力例

```
-1
```

なお，関数 `fgets` は改行も含めて入力されてしまうので注意せよ．

[1] 具体的には `strlen` や `strcat` など，名前が `str` で始まる関数を使ってはいけない．