Feladat

Gyűjtsük ki egy n elemű, szavakat tartalmazó vektorból a „G" betűvel kezdődő szavakat!

Specifikáció

A fenti feladat helyett egy általánosabb feladatot fogalmazunk meg, ráadásul vektorok helyett sorozatokra.

$$\begin{array}{rcccccc} A & = & S_{Karakter} & \times & (Szó \to \mathbb{L}) & \times & S_{Szó} \\ & & x & & \pi & & z \\ B & = & S_{Karakter} & \times & (Szó \to \mathbb{L}) & & \\ & & x' & & \pi' & & \end{array}$$

$$\begin{array}{rcl} Q & = & x' = x \wedge \pi' = \pi \\ R & = & z = seq(x'|Szó_\pi), \end{array}$$

ahol

$$\begin{array}{rcl} Betű & := & Karakter \setminus Szóköz \\ Szó & := & S_{Betű} \\ Szó_\pi & := & \{u \in Szó | \pi(u)\} \end{array}$$

Absztrakt program

A feladatot az állapottér transzformációjával oldjuk meg. Elemenkénti feldolgozással kényelmesen megoldható lenne $S_{Szó}$ beli sorozatokból kigyűjteni a π tulajdonságúakat, ezért célszerű $S_{Karakter}$-ről $S_{Szó}$-ra áttérni. A kapcsolatot x és az ílymódon kapott $y \in S_{Szó}$ között a következő szekvenciális megfelelés-sor fogja biztosítani:

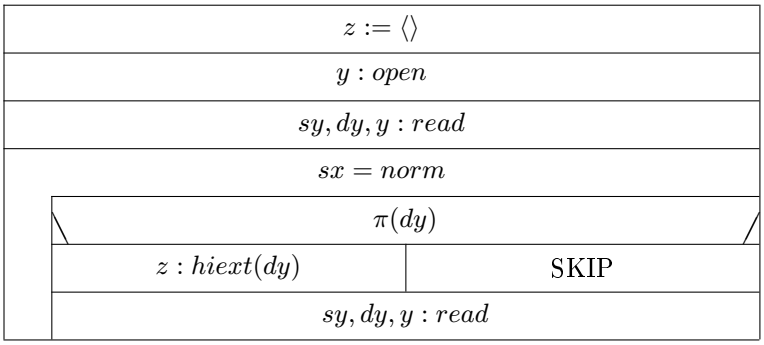
$$\begin{array}{ll} y_1 \in S_{Elem}; & seq(y_1|Karakter) = seq(x|Karakter) \\ & \forall k \in \{0, \ldots, y_1.dom - 1\} : y_{1_k}.Szó \neq y_{1_{k+1}}.Szó \end{array}$$

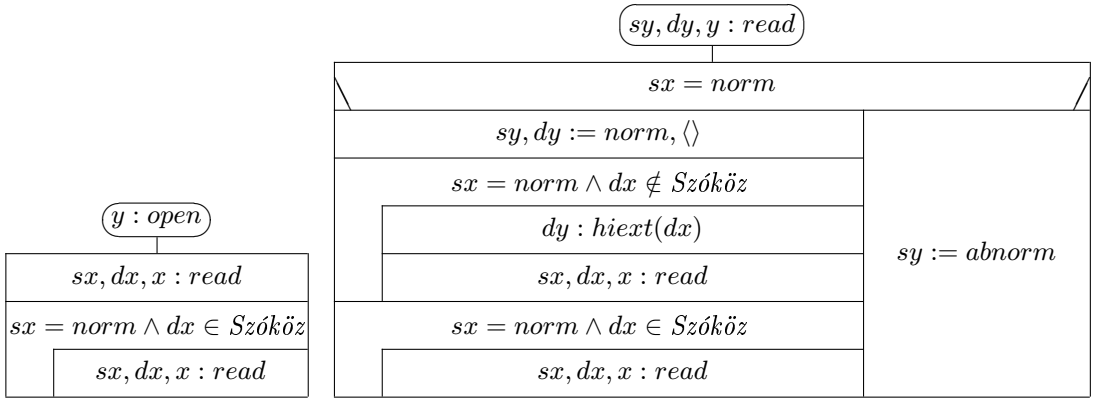
$$y \in S_{Szó}; \qquad y = seq(y_1|Szó),$$

ahol $Elem := (Szó; Elválasztó)$ és $Elválasztó := S_{Szóköz}$. Egy ilyen transzformációt követően elemenként feldolgozhatjuk y-t, az alábbi függvény segítségével:

$$f(u \in Szó) := \begin{cases} \{u\}, & \text{ha } \pi(u) \\ \emptyset, & \text{ha } \neg\pi(u) \end{cases}$$

A megfelelő, absztrakt file-ra alkalmazott elemenkénti feldolgozás tehát:





Fekete doboz-tesztelés

1. Üres sorozat
2. Szavakat nem tartalmazó (csak szóközökből álló) sorozat
3. Csak $\neg\pi$ tulajdonságú szavakat tartalmazó sorozat
4. Csak π tulajdonságú szavakat tartalmazó sorozat
5. π és $\neg\pi$ tulajdonságú szavakat egyaránt tartalmazó sorozat

C++ implementáció

A kényelmes kezelés kedvéért a fenti x, ill. y sorozatokat C++-beli osztályokként implementáljuk.

```
#include <string>
#include <list>

typedef std::string       Word;
typedef std::list<Word>   Z;
typedef bool              (Predicate) (const Word&);

enum IOState
{
    Norm,
    Abnorm
};

#include <iostream>

class X
{
    std::istream &stream;
public:
    X (std::istream &stream_);
    void Open ();
    IOState Read(char &c);
};

X::X(std::istream &stream_):
    stream(stream_)
{
```

```
}

void X::Open()
{
}

IOState X::Read(char &dx)
{
    stream.get(dx);
    return stream.eof() ? Abnorm : Norm;
}

class Y
{
    IOState  sx;
    char     dx;
    X        &x;

public:
    Y(X &x);
    void Open ();
    IOState Read(Word &dy);

private:
    static bool Whitespace(char c);
};

Y::Y(X &x_):
    x(x_)
{
    sx = x.Read(dx);
    while (sx == Norm && Whitespace(dx))
        sx = x.Read(dx);
}

void Y::Open()
{
    x.Open ();
}

IOState Y::Read(Word &dy)
{
    if (sx != Norm)
        return Abnorm;

    dy = "";
    while (sx == Norm && !Whitespace(dx))
    {
        dy += dx;
        sx = x.Read(dx);
    }

    while (sx == Norm && Whitespace(dx))
    {
        sx = x.Read(dx);
    }

    return Norm;
```

```
}

bool Y::Whitespace(char c)
{
    return (c == ' ') || (c == '\t') || (c == '\n');
}


void Filter(Y &y, Predicate pi, Z &z)
{
    z.clear ();

    IOState sy;
    Word dy;

    y.Open ();
    sy = y.Read(dy);
    while (sy == Norm)
    {
        if (pi(dy))
            z.push_back (dy);
        sy = y.Read(dy);
    }
}

bool Begins_with_G(const Word &word)
{
    return word[0] == 'G';
}

void Process (std::istream &stream)
{
    X x(stream);
    Y y(x);
    Z z;

    Filter(y, Begins_with_G, z);
    for (Z::const_iterator i = z.begin(); i != z.end(); ++i)
        std::cout << *i << std::endl;
}

#include <fstream>

void Process (const std::string &filename)
{
    if (filename == "-")
    {
        Process (std::cin);
    } else {
        std::ifstream stream (filename.c_str ());
        if (stream.fail ())
            std::cerr << "Opening " << filename << " failed." << std::endl;
        else
            Process (stream);
    }
}

int main (int argc, char **argv)
```

```
{
    if (argc > 1)
        for (int i = 1; i != argc; ++i)
            Process (argv[i]);
    else
        Process (std::cin);

    return 0;
}
```

Fehér doboz-tesztelés

1. Input utolsó sorának végén újsor-karakter szerepel/nem szerepel
2. Nemlétező file megadása