模擬国内予選2014 Problem C
壊れた暗号生成器

原案：須藤
解答：河田，森，井上，
保坂，伊藤，須藤
解説：須藤

問題概要

- 以下のような暗号を考える
  - +(文字)：次のアルファベットを表す (例: +A=B, ++A=C)
  - -(文字)：前のアルファベットを表す (例: -B=A, --C=A)
  - [(文字列)]：中身を反転した文字列を表す (例: [MCA] = ACM)
- アルファベットだった箇所がいくつか？になっている
  - 復号したとき辞書順最小になるよう？を埋めた時の
    復号後の文字列を求めよ
  - 例: J?G なら JAG, JBG, ..., JZG が有り得るので，JAG が答え
- 制約
  - 与えられる暗号文の長さは80文字以下
  - 暗号文に含まれる？の数は3個以下

解法その1

- ? の埋め方を全部試す

- 復号後の文字列のうち辞書順最小だったものを出力
  - [M+B+?] なら [M+B+A], ..., [M+B+Z]を全部復号してみる
  - BCM, ..., ZCM, ACM が得られるので，ACMを出力

- 計算量は O(暗号文の長さ*$26^{?の数}$)
  - 26はアルファベットの種類数
  - ?の数が3個なら暗号文の埋め方は 26^3 = 17576 通り
  - 暗号文の復号は文字列長の長さの線形時間で出来る(詳細は後述)
  - 暗号文の長さが最大で80, ?が3個以下なので十分間に合う

解法その2

- 暗号文を復号してから？を埋める
  - +? も -? も？として，まずは復号してしまう
- その後，?の位置を全て'A'で置換したものが答え
  - 文字の順序関係は?の埋め方によらず不変
  - よって，?の位置をAで置換すれば辞書順最小の復号結果になる
  - 例) J---?---J → J?G → JAG
- 計算量は O(暗号文の長さ)
  - "? の数が3個以下"という制約がなくても間に合う

暗号文の復号(1/4)

- 暗号文の復号はいわゆる「構文解析」の問題
  - ICPCではたびたび出題されます(簡単なものから難しいものまで)
  - 国内予選：2008年C, 2013年C
  - アジア地区予選：2009年F, 2010年J, 2011年H

- 今回のようにBNFが与えられる場合は,
  文法ごとに処理を書いていくと書きやすいです

- 次ページから実装例とともに解説します
  - グローバル変数として，暗号文を表す文字列s,
    何文字目を読んでいるかを表す整数idxが定義してあるとします

暗号文の復号(2/4)

- <Cipher> ::= <String> | <Cipher><String>
  - “Cipher は String か, CipherとString を並べたもの”という意味
  - 今回の場合は， Cipher は String を1個以上並べたものとして処理をして良い(四則演算など処理順が問題になる場合は注意)
- Cipher を読む関数の実装例

```
string readCipher(){
  string res = “”;
  // 読んでいるCipherの終端に来るまでStringを読む
  // []で囲われているCipherの場合は’]’が終端
  while(idx < s.size() && s[idx] != ‘]’){
    res += readString();
  }
  return res;
}
```

暗号文の復号(3/4)

- <String> ::= <Letter> | '['<Cipher>']‘
  - ‘[‘で始まれば [<Cipher>], そうでなければ<Letter>
- Stringを読む関数の実装例

```
string readString(){
  string res = “”;
  if(s[idx] == ‘[’){
    ++idx; // ‘[‘の分
    res = readCipher(); // []内のCipherをよむ
    reverse(res.begin(), res.end()); // 文字列の反転(C++)
    ++idx; // ‘]‘の分
  } else {
    res = readLetter(); // <Letter>を読む関数
  }
  return res;
}
```

暗号文の復号(4/4)

- <Letter> ::= ‘+’<Letter> | ‘-’<Letter> | [A-Z]
- Letterを読む関数の実装例(?の処理は省略)

```
string readLetter(){
  if(s[idx] == ‘+’){
    ++idx; // ‘+‘の分
    string res = readLetter();
    res = nextLetter(res); // resの次のアルファベットにする(実装略)
  } else if(s[idx] == ‘-’){
    // ‘+’<Letter> と同様
  } else {
    // s[idx]は大文字のアルファベット
    // idxを1進め，s[idx]の箇所のアルファベットを返す
  }
}
```

ジャッジ解

- 河田：71行(1,437B), C++
- 森　：91行(2,013B), C++
- 井上：44行(998B), C++
- 保坂：112行(2,391B), Java
- 伊藤：93行(1,854B), Java
- 須藤：42行(816B), C++