Όνομα Άσκησης: Πρώτος Κρυπτάριθμος (Prime Cryptarithm)

Πηγή: Usaco

Εκφώνηση

Ο πιο κάτω κρυπτάριθμος είναι μια εξίσωση πολλαπλασιασμού που μπορεί να λυθεί αντικαθιστώντας ψηφία από ένα σύνολο N ψηφίων στις θέσεις που σημειώνονται με αστερίσκο (*). Αν επιλεγεί ένα σύνολο ψηφίων που είναι πρώτοι αριθμοί {2,3,5,7} τότε ο κρυπτάριθμος καλείται «Πρώτος Κρυπτάριθμος».

```
   * * *
x    * *
 -------
   * * *          <-- γινόμενο 1
 * * *            <-- γινόμενο 2
 -------
 * * * *
```

Τα ψηφία μπορεί να εμφανίζονται μόνο εκεί που υπάρχουν αστερίσκοι και δεν επιτρέπεται να ξεκινούν με το μηδέν. Το γινόμενο 1 είναι το γινόμενο του πρώτου αριθμού με το δεύτερο ψηφίο του δεύτερου αριθμού, ενώ το γινόμενο 2 είναι το γινόμενο του πρώτου αριθμού με το πρώτο ψηφίο του δεύτερου αριθμού.

Να γράψετε πρόγραμμα που να βρίσκει το σύνολο των λύσεων του πιο πάνω κρυπτάριθμου για κάθε υποσύνολο ψηφίων του συνόλου {1,2,3,4,5,6,7,8,9}.

Δεδομένα εισόδου (crypt1.in)

- Ένας ακέραιος αριθμός N, ο αριθμός των ψηφίων του συνόλου
- N ψηφία με τα οποία πρέπει να λυθεί ο κρυπτάριθμος

Δεδομένα εξόδου (crypt1.out)

Ένας αριθμός με το σύνολο των λύσεων.

Παράδειγμα εισόδου

```
5
2 3 4 6 8
```

Παράδειγμα εξόδου

```
1
```

Επεξήγηση

Η μοναδική λύση του πιο πάνω παραδείγματος είναι η πιο κάτω:

```
      2 2 2    <- αριθμός a
    x   2 2    <- αριθμός b
     ------
      4 4 4    <- a * (b % 10)
    4 4 4      <- a * (b / 10)
  ---------
    4 8 8 4    <- a * b
```

Χρησιμοποιούμε δύο for loops και βρίσκουμε τα γινόμενα των a (όλοι οι τριψήφιοι) με τους b (όλοι οι διψήφιοι) και ελέγχουμε αν τα ψηφία των a, b, a * (b%10), a * (b/10) και a * b ανήκουν στο σύνολο των ψηφίων που επιτρέπονται ή όχι. Αν ναι τότε αυξάνουμε τον μετρητή μας.

Η συνάρτηση is_valid μας επιτρέπει να ελέγχουμε κάθε αριθμό αν είναι έγκυρος εφόσον αποτελείται μόνο από τα επιτρεπόμενα ψηφία.

Λύση

```
#include <fstream>
#include <set>
using namespace std;

set<int> digits;

bool is_valid(int n);

int main() {

  ifstream fin("crypt1.in");
  ofstream fout("crypt1.out");

  int N, prod, dig, g1, g2;
  int P = 0;

  fin >> N;

  for (int i = 0; i < N; i++) {
    fin >> dig;
    digits.insert(dig);
  }

  for (int a = 100; a < 1000; a++) {
    for (int b = 10; b < 100; b++) {
      if ((!is_valid(a) || (!is_valid(b))))
        continue;
      g1 = a * (b % 10);
```

```
      g2 = a * (b / 10);

      if (((g1 < 100) || (g1 > 999) || (!is_valid(g1))) || ((g2 < 100) ||
(g2 > 999) || (!is_valid(g2))))
        continue;

      dig = a * b;

      if ((dig < 1000) || (dig > 9999) || (!is_valid(dig)))
        continue;

      P++;
    }
  }

  fout << P << endl;

  fin.close();
  fout.close();

return 0;
}

bool is_valid(int n) {
  while (n) {

    if (digits.find(n % 10) == digits.end()) {
      return false;

    }
    n /= 10;
  }
  return true;
}
```