## Introduction

The Dobble game consists of 55 cards on which 8 symbols are drawn. Whatever the way of playing (there are several possible rules), the goal is always to spot as quickly as possible a common symbol between two cards. This is because two Dobble playing cards always have one and only one symbol in common.

First task (maximum number of symbols for 10 cards)

Firstly, we found the maximum number of symbols for 10 cards => 71 symbols in total


First task (minimum number of symbols for 10 cards)

We built a Dobble game with 10 cards, 13 symbols in total and 4 symbols per card. If we use the formula $n^{2}+n+1$, where $n+1$ represents the number of symbols per card ( $n=3$ ), then we get 13 cards. As we can see in the original Dobble game, it can be played with 55 cards instead of 57 cards. So you can play the game with 13 cards as well as with 10 cards.


## Second task (number of cards for 10 symbols in total)

Formulas for the relationship between the number of cards and the number of symbols:
The number of cards is $k=s+1$ and the total number of symbols is:
$n=1+2+\ldots+(s-1)+s=\frac{s(s+1)}{2}$
The maximum number of symbols $n$ is equal to the maximum number of cards $k$ :

$$
n=k=1+s(s-1)
$$

Considering the requirement of the original Dobble game, each card much have exactly 8 symbols, so we cannot build more than one card with 10 different symbols ( $8+7=15$ ). With 5,6 , or 7 symbols per card we cannot build more than one card with exactly 10 different symbols. With 4 symbols per card we can build a game with 5 cards:

$$
\begin{gathered}
s=4 \\
k=s+1=5 \\
n=\frac{s(s+1)}{2}=10
\end{gathered}
$$

Dobble game with environmentally related symbols


## Algorithm

The following algorithm outputs a set of $n^{2}+n+1$ cards that have $n+1$ symbols on each card. The cards are grouped into $n+1$ groups, by the first symbol that appears on them. The first group contains $n+1$ cards and the following contain $n$ cards each. The first group is "easy" to generate (the first card has symbols $1,2,3,4, \ldots, n+1$, the second one has symbols $1, n+2, n+3, \ldots$, $2 n+1, \ldots$ the last card has symbols $1, n^{2}+2, n^{2}+3, \ldots, n^{2}+n+1$ ). The other cards inherit the first symbol from their corresponding group, with the next symbols being generated by cyclic permutations of an identity matrix. This algorithm assumes that n is a prime number and the proof of correctness relies on some modular arithmetic observations (there exists exactly one modular inverse for each non-zero number, only if $n$ is prime).

Algorithm (C++ code)

```
#include <bits/stdc++.h>
#define NMAX 2600
#define SMAX 50
using namespace std
    int main()
    int cards [NMAX+5][SMAX+5]
    cin>>s; ///s is the number of symbols on a card
    n=s-1; ///n must be a prime number
    for(int i=1;i<=n+1;i++){
        ///generating the first group
        cards[i][1]=1
        for(int j=2;j<=s;j++){
            cards[i][j]=(i-1)*n+j
        }
    for(int i=1;i<=n;i++)
        ///generating the other groups
            for(int j=1;j<=n;j++)}
                cards[n+1+(i-1)*n+j][1]=i+1
                k<=s;k++){
                    cards[n+1+(i-1)*n+j][k]=n+1+n*(k-2)
                    +((i-1)*(k-2)+j+n-1)%n+1.
                ///cyclic permutations of symbols' indexes
        }
    for(int i=1;i<=n*n+n+1;i++){
        for(int j=1;j<=s;j++){
            cout<<cards[i][j]<<
        cout<<'\n
    return 0;
```

