Lab 3
Perfect square

```cpp
#include <iostream>
using namespace std;
int main()
{
    int a;
    int i=1;
    do{
        ...
        ...
    }while(...);
    cout<<a<<endl;
    return 0;
}
```
Perfect square

• Find and print the first perfect square (i*i) whose last two digits are both odd.

• The first perfect square I got from my program is -2147479015

• Why do I get negative number?
int

- The size of int is 4 bytes == 32 bits
- So the range of int in binary is from
  
  0000 0000 0000 0000 0000 0000 0000 0000

- to
  
  1111 1111 1111 1111 1111 1111 1111 1111

- And the first bit is sign bit, 0 means positive, 1 means negative.
- So the range of int in decimal is from -2,147,483,648 to 2,147,483,647
- Why there is one more number in negative than positive?
Binary

-2,147,483,648 = 1000 0000 0000 0000 0000 0000 0000 0000
0 = 0000 0000 0000 0000 0000 0000 0000 0000
2,147,483,647 = 0111 1111 1111 1111 1111 1111 1111 1111

So there is 0 to 2,147,483,647 are positive sign

And -2,147,483,648 to -1 are negative sign

So there is one more negative number than positive number exclude 0.
Why do I get negative number?

• The first perfect square I got from my program is -2147479015, and it computed from 46341 * 46341, and actual value for 46341 * 46341 = 2147488281

• Why 2147488281 became -2147479015?

• 2147488281 - 2147483647 = 4634

• -2147483648 + 4634 - 1 = -2147479015
Equivalent array

• Let a and b be two integer arrays of the same length. We say that they are “shift equivalent” if array a can be right shifted to create array b.

• Write a function

    ```
    bool equivalent(int a[], int b[], int n)
    ```

• Which takes two arrays a and b of length n and returns `true` if they are shift equivalent and `false` otherwise.

• How do you approach this problem?
Logic Flow

• Step 1: check is array a equal to array b?
  • yes -> Finish, they are equivalent
  • no -> jump to Step 2

• Step 2: right shift array a

• Step 3: is the right shifted array same as original array a
  • yes -> Finish, they are not equivalent
  • no -> jump to Step 1
#include <iostream>
using namespace std;

void right_shift(int a[], int size)
{
    ...
    ...
}

bool equivalent(int a[], int b[], int size)
{
    ...
    ...
}

int main()
{
    int a[5] = {1, 2, 3, 4, 5};
    int b[5] = {2, 3, 4, 5, 1};
    bool isSame = false;
    for (int i = 0; i < 5; i++)
    {
        if (equivalent(a, b, 5))
            isSame = true;
        right_shift(a, 5);
    }
    if (isSame)
        cout << "a and b are equivalent." << endl;
    else cout << "a and b are not equivalent." << endl;
    return 0;
}
goto

• Using goto as loop:

Label:

    //statements
    goto Label;

• Using goto as branching(if):

    //statements
    if (true) goto Label;
    //more statement

Label:
```cpp
#include <iostream>
using namespace std;

int main()
{
    int input;

    s1:
    cout << "s1 step: please re input 1 or 0" << endl;
    cin >> input;
    if (input == 1)
        goto dead;
    else if (input == 0)
        goto s2;

    s2:
    cout << "s2 step: please re input 1 or 0" << endl;
    cin >> input;
    if (input == 1)
        goto s2;
    else if (input == 0)
        goto s1;

    dead:
    cout << " all of program is done." << endl;
    return 0;
}
```
Eight Queens

• The eight queens puzzle is the problem of placing eight chess queens on an 8x8 chessboard so that no two queens threaten each other. Thus, a solution requires that no two queens share the same row, column, or diagonal.

• Now, how can we compute the number of possible positions to place the Queens to the chessboard?