Problem G. Second maximum - 2

The sequence consists of natural numbers and ends with the number 0.
Determine the value of the second largest element in this sequence, that is, the element that will be largest if you remove the largest element from the sequence.
Input data
A sequence of integers is introduced, ending with the number 0 (the number 0 itself is not included in the sequence, but serves as a sign of its end).

Output
Print the answer to the problem.

Examples of input data
one
7
9
0

output
7

input data
2
1
0

output
1

```
void secondMaximum() {
  int number;
  int maxNum = 0;
  int secondMax = 0;
  while(1) {
     std::cin >> number;
     if (number == 0) {
        break;
     }
     else {
        if (number > maxNum) {
           secondMax = maxNum;
           maxNum = number;
        }
        else if (number > secondMax) {
           secondMax = number;
        }
     }
  }
  std::cout << "Max number is: " << maxNum << std::endl;
  std::cout << "Second max number is: " << secondMax << std::endl;
}
```

hw: check how done with array, should be similar

```
while(1)
{
    cin>>....
    if(zero)
    {
        break;
    }
}
```

2nd version:
how to emulate break with status variable?

```
void secondMaximum() {
 int number;
 int maxNum;
 int flag = 0;
 int secondMax;
 int thirdMax;
 while(1) {
     std::cin >> number;
    if (number == 0) {
       break;
    }
    else {
        if (flag == 0) {
            maxNum = number;
            flag = 1;
        }
        else if (flag == 1) {
           if (number > maxNum) {
               secondMax = maxNum;
               maxNum = number;
           }
           else {
               secondMax = number;
           }
           flag = 2;
        }
        else if (flag >= 2) {
           if (number > maxNum) {
               thirdMax = secondMax;
               secondMax = maxNum;
               maxNum = number;
           }
           else if (number > secondMax) {
               thirdMax = secondMax;
               secondMax = number;
           }
           else if (flag == 2){
               thirdMax = number;
           }
           else if (flag == 3 && thirdMax > number) {
               thirdMax = number;
           }
           flag = 3;
        }
    }
 }
 std::cout << flag << std::endl;
 std::cout << "Max number is: " << maxNum << std::endl;
 std::cout << "Second max number is: " << secondMax << std::endl;
 std::cout << "Third max number is: " << thirdMax << std::endl;
}
```

find fifth maximum by value in stream of numbers (1000 000 000 numbers)

1st way (small stream)
collect all numbers in array and sort array

2st way (any stream)
5,1,7,3,5,0,-2,11,18,8,9
ar=[]
[5]
[5,1]->[1,5]
[1,5,7]
[1,5,7,3]->[1,3,5,7]
[1,3,5,7,0]->[0,1,3,5,7]
[0,1,3,5,7,-2]->[-2,0,1,3,5,7]->[0,1,3,5,7]
[0,1,3,5,7,11]->[1,3,5,7,11]
[1,3,5,7,11,18]->[3,5,7,11,18]
[3,5,7,11,18,8]->[3,5,7,8,11,18]->[5,7,8,11,18]
[5,7,8,11,18,9]->[5,7,8,9,11,18]->[7,8,9,11,18]

[-2,0,1,3,5,7]->[0,1,3,5,7,7]
[0,1,3,5,7,11]

```
void secondMaximum() {
 int number;
 int flag;
 int order = 7;
 int temp;
 int k;
 vector <int> myHero;
 while(1) {
     std::cin >> number;
    if (number == 0) {
       break;
    }
    else {
       flag = 0; // if we haven't encountered a number
       for (int i = 0; i < myHero.size(); i++) {
           if (number == myHero[i]) {
               flag = 1; // if we encounter first number
               break;
           }
       }
       if (flag == 0) {
           if (myHero.size() < order) {
               myHero.push_back(number);
               k = myHero.size() - 1;
               while (k > 0 && myHero[k] < myHero[k - 1]) {
                   // This process starts the ascending order
                   temp = myHero[k];
                   myHero[k] = myHero[k - 1];
                   myHero[k - 1] = temp;
                   k--;
               }
           }
           else {
               for (int i = 0; i < myHero.size() - 1; i++) {
                   myHero[i] = myHero[i+1];
               }
               myHero[myHero.size() - 1] = number;
               k = myHero.size() - 1;
               while (k > 0 && myHero[k] < myHero[k - 1]) {
                   // This process starts the ascending order
                   temp = myHero[k];
                   myHero[k] = myHero[k - 1];
                   myHero[k - 1] = temp;
                   k--;
               }
           }
       }

    }
 }
 std::cout << myHero[0] << std::endl;
 for (int i = 0; i < myHero.size(); i++) {
     cout << myHero[i] << " ";
 }
}
```