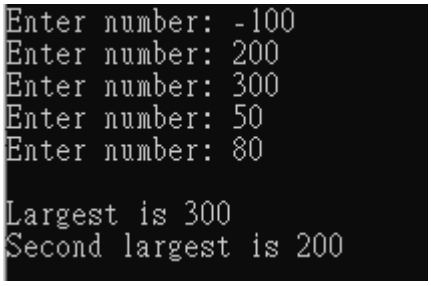
1. (***Find the two Largest Numbers***) Write a C# app that inputs a series of **five** integers, then determines and displays the **two** largest integers. (Exercise 5.23, P. 227)

   Possible output:



2. (***Account Modification***) Modify class Account (Fig. 4.11) to provide a Withdraw method that withdraws money from an Account. Ensure that the withdrawal amount doesn't exceed the balance. If it does, the balance should not be changed and the method should display a message indicating "Withdrawal amount exceeded account balance." Modify class AccountTest (Fig. 4.12) to test method Withdraw. (Exercise 4.9, P. 180)

   Some statements in AccountTest:

```
Account account1 = new Account("Tom Bruice", 30.55m);
Account account2 = new Account("John Wayne", -20.88m);
```

   Some statements in Account:

```
public void Withdraw(decimal withdrawalAmount)
{
   if (withdrawalAmount > Balance)
```

   Possible output:

```
Tom Bruice's balance: NT$30.55
John Wayne's balance: NT$0.00

Enter deposit amount for account1: 100
adding NT$100.00 to account1 balance

Tom Bruice's balance: NT$130.55
John Wayne's balance: NT$0.00

Enter deposit amount for account2: 200
adding NT$200.00 to account2 balance

Tom Bruice's balance: NT$130.55
John Wayne's balance: NT$200.00
Enter withdrawal amount for account1: 50
subtracting NT$50.00 from account1 balance

Tom Bruice's balance: NT$80.55
John Wayne's balance: NT$200.00
Enter withdrawal amount for account2: 400
subtracting NT$400.00 from account2 balance

Withdrawal amount exceeded account balance.
Tom Bruice's balance: NT$80.55
John Wayne's balance: NT$200.00
```

3. (***Student Record Class***) Create a class called Student that an institute might use to represent a record for students qualifying from the institute. A Student record should include **four** pieces of information as either instance variables or auto-implemented properties: a student's id (type string), a student's name (type string), and **two** separate variables for scores in **two** subjects (type decimal). Your class should have a constructor that initializes the **four** values. Provide a property with a get and set accessor for any instance variables. For the scores in subjects, if the value passed to the set accessor is negative, the value of the instance variable should be left unchanged. Also, provide methods named GetAggregate and GetPercentage that calculate the aggregate marks in the **two** subjects (sum of **two** subject marks) and the percentage (i.e., sum divided by the maximum marks, **120** (not 100, each subject), and then multiplied by 100), and then return the aggregate and percentage as decimal value. Write a test app named StudentRecordTest that demonstrates class Student's capabilities. (Exercise 4.10, P. 180)

Some statements in StudentRecordTest:

```
// change Student1's data
stud1.StudentID = "A110";
stud1.PSub1 = 120;
```

```
// change Student2's data
stud2.StudentID = "A220";
stud2.PSub2 = 105;
```

Possible output:

```
Student information
Student ID: A01
Student Name: Mary
Subject1: 60
Subject2: 90
Aggregate: 150
Percentage: 62.50

Student with changed information
Student ID: A110
Student Name: Mary
Subject1: 120
Subject2: 90
Aggregate: 210
Percentage: 87.50

Original Student information
Student ID: A02
Student Name: John
Subject1: 80
Subject2: 70
Aggregate: 150
Percentage: 62.50

Updated Student information
Student ID: A220
Student Name: John
Subject1: 80
Subject2: 105
Aggregate: 185
Percentage: 77.08
```

4. Modify the app in Fig. 6.5. First, enter the initial value and the final value for for-loop. Then, calculate the sums of odd, even and all numbers. Finally, print them on the screen. (Fig. 6.5, PP. 239–240)

   Possible inputs and outputs:

```
Enter initial value for for-loop: 1
Enter final value for for-loop: 100
Sum of odd numbers is 2500
Sum of even numbers is 2550
Sum of all numbers is 5050
```

```
Enter initial value for for-loop: 10
Enter final value for for-loop: 20
Sum of odd numbers is 75
Sum of even numbers is 90
Sum of all numbers is 165
```

5. Modify the app in Fig. 6.14. First, randomly generate a number. Make the count variable equal to the number. Finally, print all integers from 1 to 10 excluding the skipped number on the screen. (Fig. 6.14, P. 255)

   Some statements must be written in Main(). You can see Lines 9 and 21 at P. 295 for reference.

   Next(int, int): Returns a positive random integer within the specified minimum and maximum range (includes min and excludes max).

   Possible inputs and outputs: