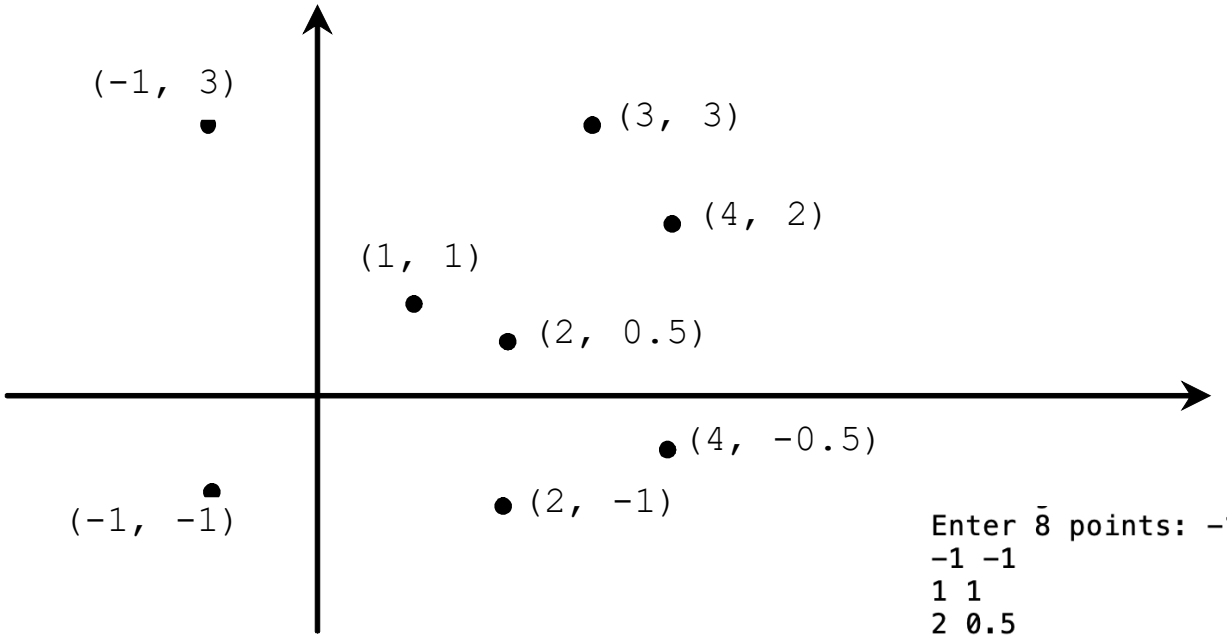
PROGRAMMING WITH C++

LAB 8

Xi'an Jiaotong-Liverpool University
西交利物浦大学

Example 1: Finding Two Points Nearest to Each Other

Given a set of points, the closest-pair problem is to find the two points that are nearest to each other. In Figure below, for example, points (1, 1) and (2, 0.5) are closest to each other.



	x	y
0	-1	3
1	-1	-1
2	1	1
3	2	0.5
4	2	-1
5	3	3
6	4	2
7	4	-0.5

```
Enter 8 points: -1 3
-1 -1
1 1
2 0.5
2 -1
3 3
4 2
4 -0.5
The closest two points are (1, 1) and (2, 0.5)
```

Example 2: Daily Temperature and Humidity

Suppose a meteorology station records the temperature and humidity at each hour of every day and stores the data for the past ten days in a text file named weather.txt. Each line of the file consists of four numbers that indicates the day, hour, temperature, and humidity. The contents of the file may look like the one in (a):

```
1 1 76.4 0.92
1 2 77.7 0.93
...
10 23 97.7 0.71
10 24 98.7 0.74
```

(a)

```
10 24 98.7 0.74
1 2 77.7 0.93
...
10 23 97.7 0.71
1 1 76.4 0.92
```

(b)

Your task is to write a program that calculates the average daily temperature and humidity for the 10 days.

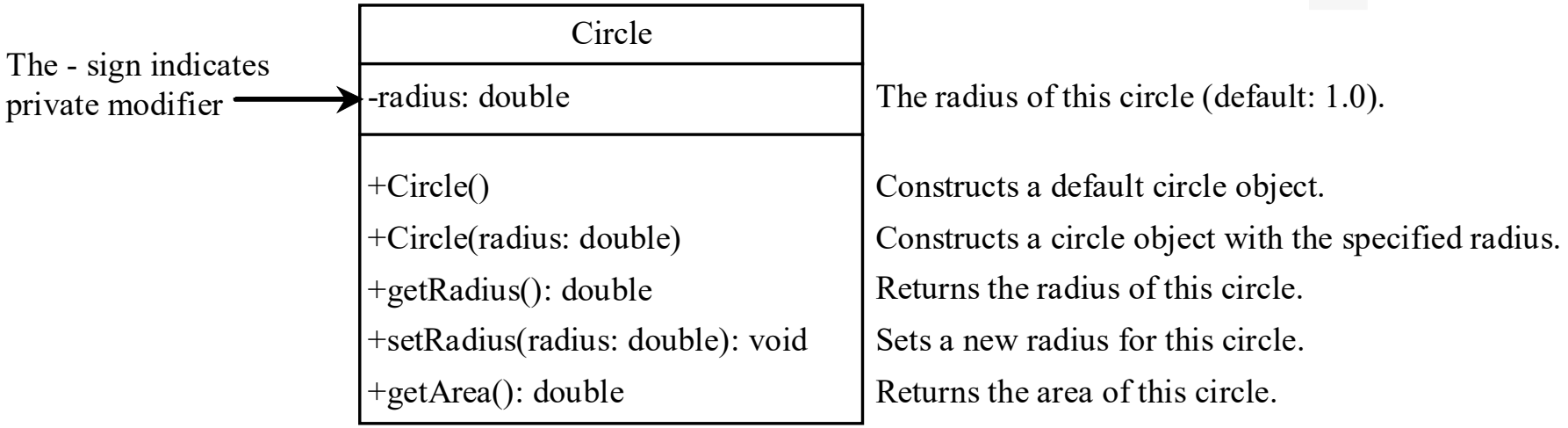
Example 3: Defining Classes And Creating Objects

As the example, consider TV sets. Each TV is an object with state (current channel, current volume level, power on or off) and behaviors (change channels, adjust volume, turn on/off). You can use a class to model TV sets. The UML diagram for the class is shown below.

TV	
channel: int	The current channel (1 to 120) of this TV.
volumeLevel: int	The current volume level (1 to 7) of this TV.
on: bool	Indicates whether this TV is on/off.
+TV()	Constructs a default TV object.
+turnOn(): void	Turns on this TV.
+turnOff(): void	Turns off this TV.
+setChannel(newChannel: int): void	Sets a new channel for this TV.
+setVolume(newVolumeLeve: int): void	Sets a new volume level for this TV.
+channelUp(): void	Increases the channel number by 1.
+channelDown(): void	Decreases the channel number by 1.
+volumeUp(): void	Increases the volume level by 1.
+volumeDown(): void	Decreases the volume level by 1.

Example 4: New Circle Class

Let us create a new circle class with a private data field radius and its associated accessor and mutator functions. The class diagram is shown in Figure below.



Example 5: Sudoku

5	3			7				
6			1	9	5			
	9	8					6	
8				6				3
4			8		3			1
7				2				6
	6							
			4	1	9			5
				8			7	9

5	3	*4*	*6*	7	*8*	*9*	*1*	*2*
6	*7*	*2*	1	9	5	*3*	*4*	*8*
1	9	8	*3*	*4*	*2*	*5*	6	*7*
8	*5*	*9*	*7*	6	*1*	*4*	*2*	3
4	*2*	*6*	8	*5*	3	*7*	*9*	1
7	*1*	*3*	*9*	2	*4*	*8*	*5*	6
9	6	*1*	*5*	*3*	*7*	*2*	*8*	*4*
2	*8*	*7*	4	1	9	*6*	*3*	5
3	*4*	*5*	*2*	8	*6*	*1*	7	9

The objective is to fill the grid (see the left figure) so that every row, every column, and every 3×3 box contain the numbers 1 to 9, as shown in the right figure.

Example 5: Sudoku

5	3			7				
6			1	9	5			
	9	8					6	
8				6				3
4			8		3			1
7				2				6
	6							
			4	1	9			5
				8			7	9

5	3	1	2	7	4	8	9	
6			1	9	5			
	9	8					6	
8				6				3
4			8		3			1
7				2				6
	6							
			4	1	9			5
				8			7	9

5	3	1	2	7	4	9		
6			1	9	5			
	9	8					6	
8				6				3
4			8		3			1
7				2				6
	6							
			4	1	9			5
				8			7	9

Rule 1: Fill in an empty cell from the first to the last.

Rule 2: Fill in a smallest number possible.

Rule 3: If no number can fill in a cell, backtrack.

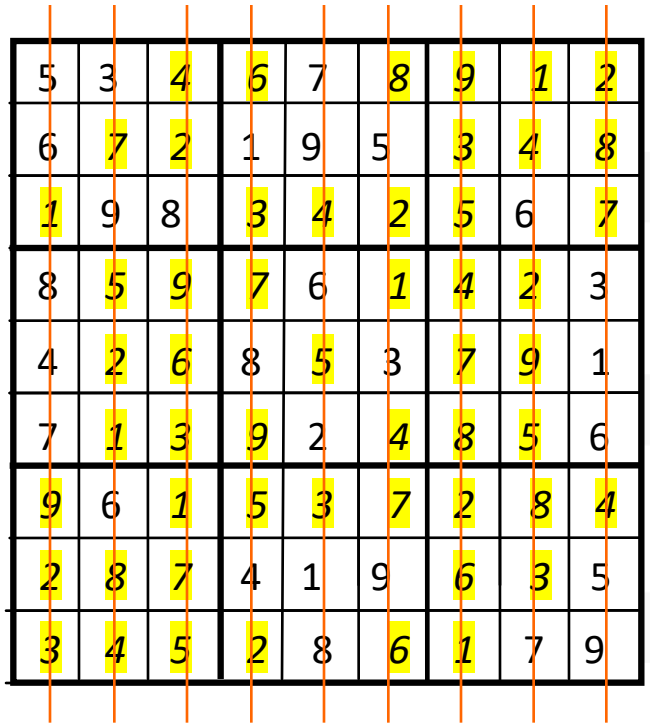
Every row contains the numbers 1 to 9

5	3			7				
6			1	9	5			
	9	8					6	
8				6				3
4			8		3			1
7				2				6
	6							
			4	1	9			5
				8			7	9

5	3	*4*	*6*	7	*8*	*9*	*1*	*2*
6	*7*	*2*	1	9	5	*3*	*4*	*8*
1	9	8	*3*	*4*	*2*	*5*	6	*7*
8	*5*	*9*	*7*	6	*1*	*4*	*2*	3
4	*2*	*6*	8	*5*	3	*7*	*9*	1
7	*1*	*3*	*9*	2	*4*	*8*	*5*	6
9	6	*1*	*5*	*3*	*7*	*2*	*8*	*4*
2	*8*	*7*	4	1	9	*6*	*3*	5
3	*4*	*5*	*2*	8	*6*	*1*	7	9

Every column contains the numbers 1 to 9

5	3			7				
6			1	9	5			
	9	8					6	
8				6				3
4			8		3			1
7				2				6
	6							
			4	1	9			5
				8			7	9



Every 3×3 box contains the numbers 1 to 9

5	3			7				
6			1	9	5			
	9	8					6	
8				6				3
4			8		3			1
7				2				6
	6							
			4	1	9			5
				8			7	9

5	3	*4*	*6*	7	*8*	*9*	*1*	*2*
6	*7*	*2*	1	9	5	*3*	*4*	*8*
1	9	8	*3*	*4*	*2*	*5*	6	*7*
8	*5*	*9*	*7*	6	*1*	*4*	*2*	3
4	*2*	*6*	8	*5*	3	*7*	*9*	1
7	*1*	*3*	*9*	2	*4*	*8*	*5*	6
9	6	*1*	*5*	*3*	*7*	*2*	*8*	*4*
2	*8*	*7*	4	1	9	*6*	*3*	5
3	*4*	*5*	*2*	8	*6*	*1*	7	9

Checking Whether a Solution Is Correct

5	3			7				
6			1	9	5			
	9	8					6	
8				6				3
4			8		3			1
7				2				6
	6							
			4	1	9			5
				8			7	9

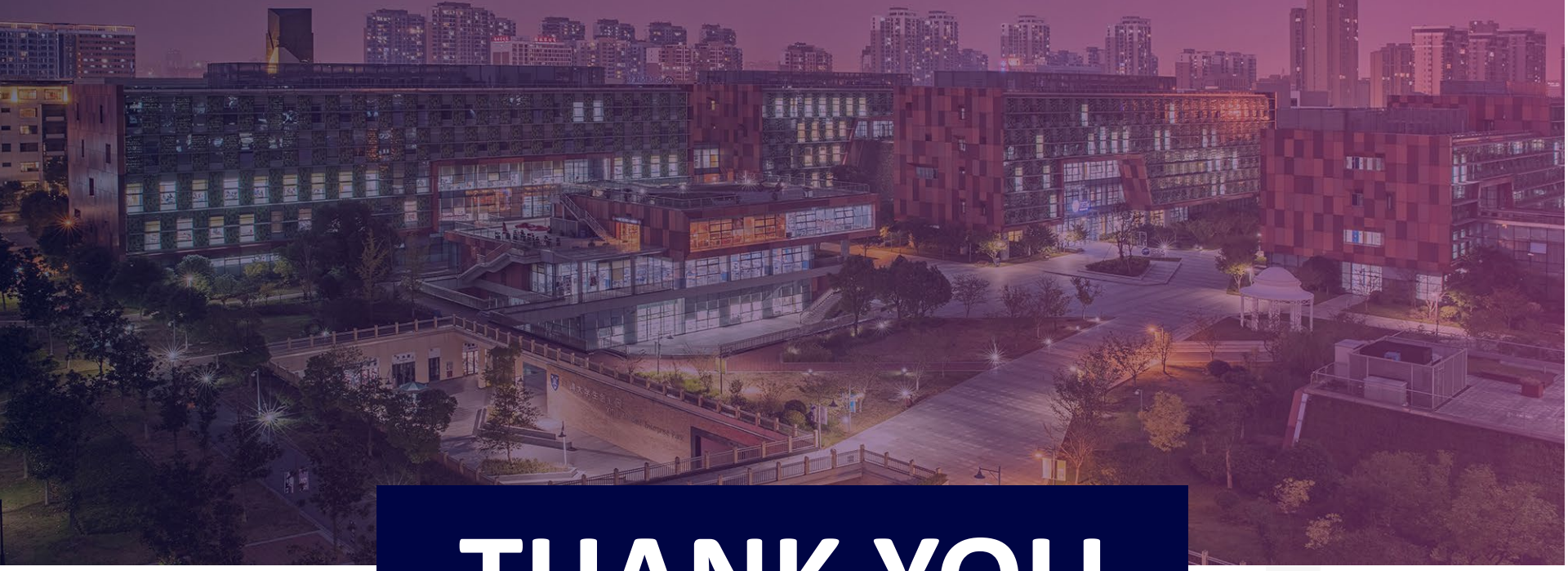
5	3	*4*	*6*	7	*8*	*9*	*1*	*2*
6	*7*	*2*	1	9	5	*3*	*4*	*8*
1	9	8	*3*	*4*	*2*	*5*	6	*7*
8	*5*	*9*	*7*	6	*1*	*4*	*2*	3
4	*2*	*6*	8	*5*	3	*7*	*9*	1
7	*1*	*3*	*9*	2	*4*	*8*	*5*	6
9	6	*1*	*5*	*3*	*7*	*2*	*8*	*4*
2	*8*	*7*	4	1	9	*6*	*3*	5
3	*4*	*5*	*2*	8	*6*	*1*	7	9

Enter a Sudoku puzzle solution:
9 6 3 1 7 4 2 5 8
1 7 8 3 2 5 6 4 9
2 5 4 6 8 9 7 3 1
8 2 1 4 3 7 5 9 6
4 9 6 8 5 2 3 1 7
7 3 5 9 6 1 8 2 4
5 8 9 7 1 3 4 6 2
3 1 7 2 4 6 9 8 5
6 4 2 5 9 8 1 7 3
Valid solution

CheckSudokuSolution.cpp

```
#include <iostream>
using namespace std;

void readASolution(int grid[][9]);
bool isValid(const int grid[][9]);
bool isValid(int i, int j, const int grid[][9]);

int main()
{
  // Read a Sudoku puzzle
  int grid[9][9];
  readASolution(grid);

  cout << (isValid(grid) ? "Valid solution" : "Invalid
solution");

  return 0;
}

/** Read a Sudoku puzzle from the keyboard */
void readASolution(int grid[][9])
{
  cout << "Enter a Sudoku puzzle:" << endl;
  for (int i = 0; i < 9; i++)
    for (int j = 0; j < 9; j++)
      cin >> grid[i][j];
}
```

```
// Check whether the fixed cells are valid in the grid
bool isValid(const int grid[][9])
{
  for (int i = 0; i < 9; i++)
    for (int j = 0; j < 9; j++)
      if (grid[i][j] < 1 || grid[i][j] > 9 ||
          !isValid(i, j, grid))
        return false;
  return true; // The fixed cells are valid
}
// Check whether grid[i][j] is valid in the grid
bool isValid(int i, int j, const int grid[][9])
{
  // Check whether grid[i][j] is valid at the i's row
  for (int column = 0; column < 9; column++)
    if (column != j && grid[i][column] == grid[i][j])
      return false;
  // Check whether grid[i][j] is valid at the j's column
  for (int row = 0; row < 9; row++)
    if (row != i && grid[row][j] == grid[i][j])
      return false;
  // Check whether grid[i][j] is valid in the 3-by-3 box
  for (int row = (i / 3) * 3; row < (i / 3) * 3 + 3; row++)
    for (int col = (j / 3) * 3; col < (j / 3) * 3 + 3; col++)
      if (row != i && col != j && grid[row][col] == grid[i][j])
        return false;
  return true; // The current value at grid[i][j] is valid
}
```



THANK YOU

VISIT US

WWW.XJTLU.EDU.CN

FOLLOW US

Xi'an Jiaotong-Liverpool University
西交利物浦大学