Week 6: In-Class Exercises

***Please download the resource from** http://blue.smu.edu.sg/cs101/ice6-resource.zip*

1. You are given **q1.c**. Implement the **calculate_ticket_price** function. The function takes in 1 parameter: age (type: `int`). The ticket price is as follows:

1. Adult (age 13 - 59):	$76
2. Child (age 4 - 12):	$56
3. Senior (age 60 and above):	$38
4. Children below 4	Free ($0)

   You can assume the user enters a valid age.

   e.g. 1: If the method is invoked like this:

```
printf("%d\n", calculate_ticket_price(13));
```

   the statement generates the following output:

```
76
```

2. You are given **q2.c**. Implement the `gcd` function. The function takes in two parameters: `num1`(type: `int`) and `num2` (type: `int`) and returns the greatest common divisor of `num1` and `num2`. The greatest common divisor is simply the biggest number that divides each of the integers without leaving a remainder. This method returns 0 if both parameters are 0.

   e.g. 1: If the method is invoked like this:

```
printf("%d\n", gcd(9, 12));
```

   the statement generates the following output:

```
3
```

   e.g. 2: If the method is invoked like this:

```
printf("%d\n", gcd(-3,-6));
```

   the statement generates the following output:

```
3
```

   Note: GCD(a,b) = G(|a|, |b|)

3. Write a function `display_ordinal_number` that takes in a parameter number (type: int). If the number is greater than or equal to 0, this function returns a string of the number followed by the ordinal suffix ('st', 'nd', 'rd', 'th'). For example,
    a. `display_ordinal_number(0)` will return the value `'0th'`
    b. `display_ordinal_number(1)` will return the value `'1st'`.
    c. `display_ordinal_number(2)` will return the value `'2nd'`.
    d. `display_ordinal_number(3)` will return the value `'3rd'`.
    e. `display_ordinal_number(4)` will return the value `'4th'`.

    Otherwise, the function will just return the original value without any suffix.

    The logic to decide on the ordinal suffix is as follows:

Number ends with	**Suffix**
11, 12, or 13	th
1	st
2	nd
3	rd
Everything else	th

4. Implement the **print_expanded** function in **q4.c**. This function takes in a positive integer, and prints the number in expanded form.

    e.g. 1: If the method is invoked like this:
    `print_expanded_form(123);`

    the statement generates the following output:

```
3 + 20 + 100
```

    e.g. 2: If the method is invoked like this:
    `print_expanded_form(-163)`

the statement generates the following output:

```
Invalid Input!
```

5. Implement the **get_smallest_pair** function in **q5.c** When **get_smallest_pair** method is invoked with **num**, it returns the integer which is the SMALLEST of all possible pairs of 2 consecutive digits formed from **num**.
   Note: You can assume **num** is a positive number.

   For example,
   1. if n is 2345, the possible pair of digits are: 23, 34, 45. The smallest pair is 23
   2. if n is 10245, the possible pair of digits are 10, 2 (formed because of 02),24, and 45. The smallest pair is 2.

number	Possible Pairs	Positions in number
2345	23 34 45	**23**45 2**34**5 23**45**
10245	10 2 24 45	**10**245 1**02**45 10**24**5 102**45**

e.g. 1: If the method is invoked like this:

```
print(get_smallest_pair(2345))
```

the statement generates the following output:

```
23
```

e.g. 2: If the method is invoked like this:

```
print(get_smallest_pair(10245))
```

the statement generates the following output:

```
2
```

e.g. 2: If the method is invoked like this:

```
print(get_smallest_pair(2))
```

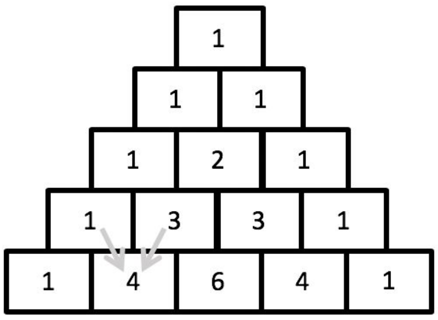
the statement generates the following output:

```
-1
```

Note:
1. If the input parameter does not contain 2 consecutive digits, return the value of -1.

6. **[Difficulty: ***]** Implement the **print_pascal** function in **q6.c** and display Pascal's triangle.

   To build the triangle, start with "1" at the top, then continue placing numbers below it in a triangular pattern. Each number is the numbers directly above it added together.



   e.g. 1: If the method is invoked like this:
   `print(print_pascal(5))`

   the statement generates the following output:

```
1
1 1
1 2 1
1 3 3 1
1 4 6 4 1
```

7. **[Difficulty: **]** A sorting algorithm is an algorithm made up of a series of instructions that takes a list as input, performs specified operations on the list, and outputs a sorted list. This topic will be covered in depth in another course called "Data Structures and Algorithms".

   Bubble sort is one of the simplest sorting algorithms that compares each pair of elements in a list and swaps them if they are out of order until the entire list is sorted. For each element in the list, the algorithm compares every pair of elements.

   Assume that you have the array {7, 3, 6, 5, 1}. In the first iteration, the adjacent elements are being compared to see if they are out of order. If there are n items in the list, then there are (n - 1) comparisons.

First Pass

Note: The highlighted cells show the two values being compared.

7	3	6	5	1

(7 is greater than 3. Swap the values)

3	7	6	5	1

(7 is greater than 6. Swap the values)

3	6	7	5	1

(7 is greater than 5. Swap the values)

3	6	5	7	1

(7 is greater than 1. Swap the values)

3	6	5	1	**7**

At the end of the first iteration, the largest value(i.e. 7) is at the correct position.

Second Pass

3	6	5	1	**7**

(3 is smaller than 1. Do nothing.)

3	6	5	1	**7**

(6 is larger than 5. Swap the values.)

3	5	6	1	**7**

(6 is larger than 1. Swap the values.)

3	5	1	**6**	**7**

At the end of the second iteration, the 2nd largest value(i.e. 6) is at the correct position.

Third Pass

3	5	1	**6**	**7**

(3 is smaller than 5. No exchange of values)

3	5	1	**6**	**7**

(5 is larger than 1. Swap the values.)

3	1	**5**	**6**	**7**

At the end of the third iteration, the 3nd largest value(i.e. 5) is at the correct position.

Fourth Pass

3	1	**5**	**6**	**7**

(3 is larger than 1. Swap the values.)

1	**3**	**5**	**6**	**7**

At the end of the fourth iteration, 1 & 3 are both at the correct positions.

Implement the `bubble_sort()` function.

Reference:

1. https://www.toptal.com/developers/sorting-algorithms: You can watch the animation here to compare the efficiency of various sorting algorithms.