- Username : class-1
- Password :

- Check VS 2019 whether can use
- We will start our course in 18:30
- we will start demonstrate the exercises at 19:15.
- Do not use scanf_s
- Please make sure the TA has recorded your exercise score here before leaving.

Schedule

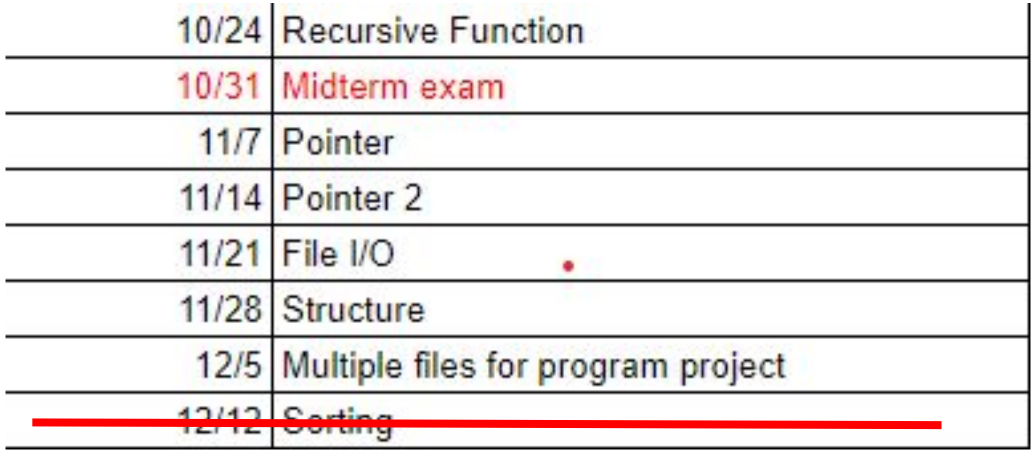
12/15: Week12 homework late submission deadline
12/16: Upload sample code on E3(sample code on drive)
12/19: Final exam
12/26: No class
1/2 : Review & practice (Late demo for this week exercise)
1/9 : The third exam

Note

- We will not provide the sample code for week13 homework. For this topic(multiple file), we will upload sample code for the exercises.
- If you have any questions, you can email the TAs to schedule a meeting.

Final Exam

Final will cover the content of the TA course from 10/24 to 12/5.



10/24	Recursive Function
10/31	Midterm exam
11/7	Pointer
11/14	Pointer 2
11/21	File I/O
11/28	Structure
12/5	Multiple files for program project
~~12/12~~	~~Sorting~~

Final Exam Rules

- Final will be held on 12/19 from 18:30 to 21:40.
- You must use the classroom computers.
- We will provide paper exam sheets, and you can bring paper and pencils, but the sheets should be two blank sides.
- Smart watches and phones must be placed inside your bags.
- You cannot use any library in C++, or you will get score 0.
- The final exam will start on 18:30, you must log into new E3 from 18:30 to 18:40, after 18:40 you are not allowed to use internet except new E3 (you cannot log in since you need to log into NYCU portal first, which is prohibited).
- You can download anything on new E3, like slides, homework, exercise and sample code(we will upload tomorrow).
- We will give you the test paper on 18:40, then you can start writing.
- Cheating will be 0 score and be punished.

Final Exam Rules

- The network will be restored at 21:35. Please upload your files to E3 before 21:40.
- We will note in the problem whether it will be graded using OJ or Visual Studio. If it’s graded using Visual Studio, such as file I/O or multiple file, we will also provide OJ for you, you can refer to the results on the OJ.

Sorting

Introduction to Computers and Programming
Week14 TA Course
2023/12/12

What is sorting?

- Sorting is an algorithm that puts elements of a list in a certain order.
- Example

  Before sort : 3, 5, 19, 1, 3, 10

  After sort : 1, 3, 3, 5, 10, 19

Types of Sorting

- Comparison based Sorting
  - Only reads the list elements through a single abstract comparison operation (often a "less than or equal to" operator) that determines which of two elements should occur first in the final sorted list.
    - Iterative sorting algorithms : Selection Sort, Bubble Sort, Insertion Sort
    - Recursive sorting algorithms : Merge Sort, Quick Sort
- Non-Comparison based Sorting
  - Don't use comparison but rely on integer arithmetic on keys.
    - Radix sort

Stable sorting

- A sorting algorithm is stable if the relative order of elements with the same key value is preserved by the algorithm.
- Example :

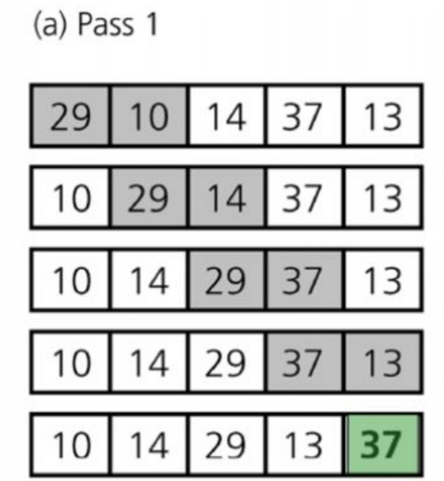
  Given a list of number : 3, 5, 19, 1, 3*, 10

  Stable sorting : 1, 3, 3*, 5, 10, 19

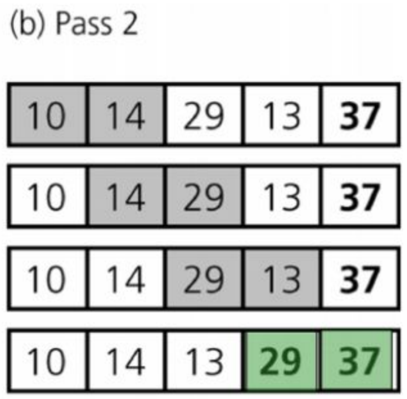
  Non-stable sorting : 1, 3*, 3, 5, 10, 19

Bubble Sort

- Idea : The larger items drop down while the smaller ones bubble up.
- Property : Stable



At the end of **Pass 1**, the largest item **37** is at the last position.

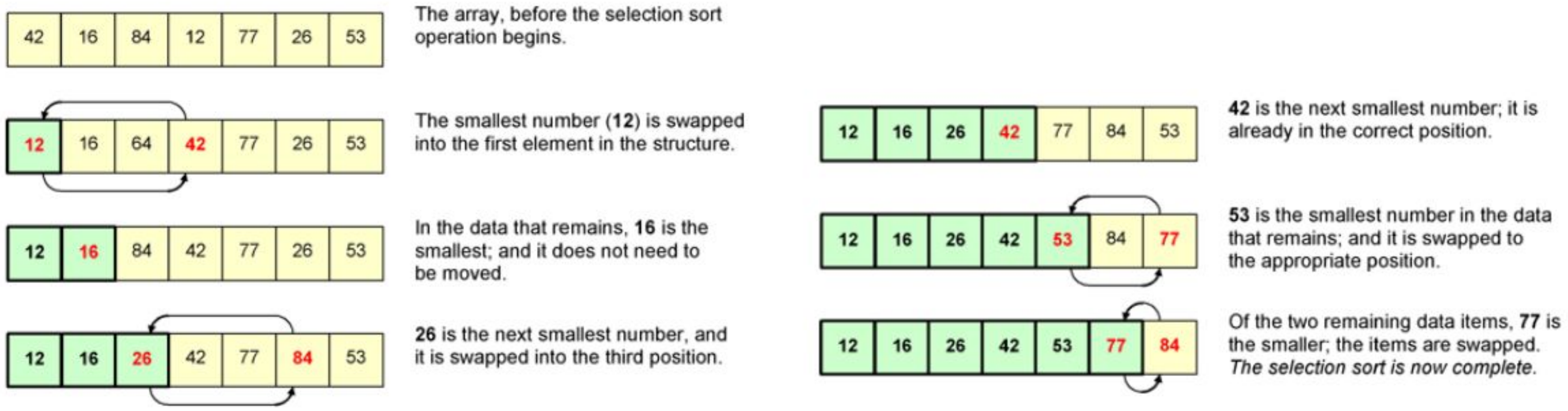


At the end of **Pass 2**, the second largest item **29** is at the second last position.

🟩 x	**Sorted Item**
▨ x	**Pair of items under comparison**

Selection Sort

- Idea : Selecting the smallest element from the unsorted portion of the list and moving it to the sorted portion of the list.



The array, before the selection sort operation begins.

The smallest number (**12**) is swapped into the first element in the structure.

In the data that remains, **16** is the smallest; and it does not need to be moved.

26 is the next smallest number, and it is swapped into the third position.

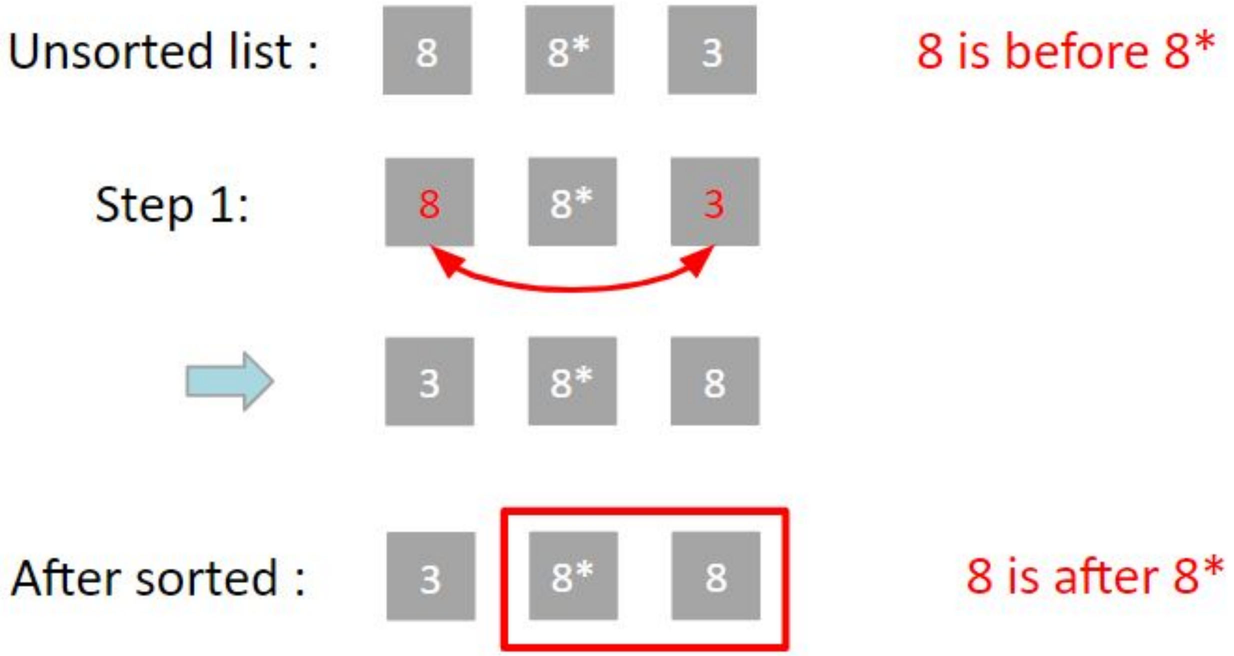
42 is the next smallest number; it is already in the correct position.

53 is the smallest number in the data that remains; and it is swapped to the appropriate position.

Of the two remaining data items, **77** is the smaller; the items are swapped. *The selection sort is now complete.*

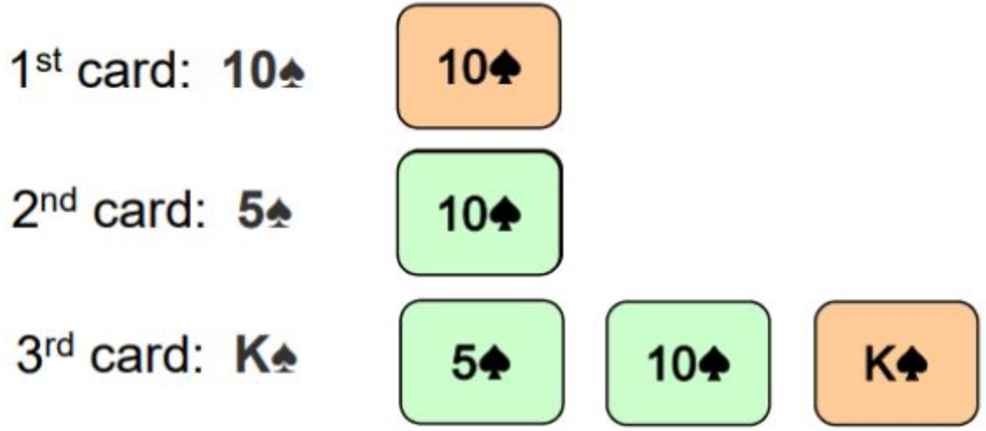
Selection Sort

- Property : Unstable

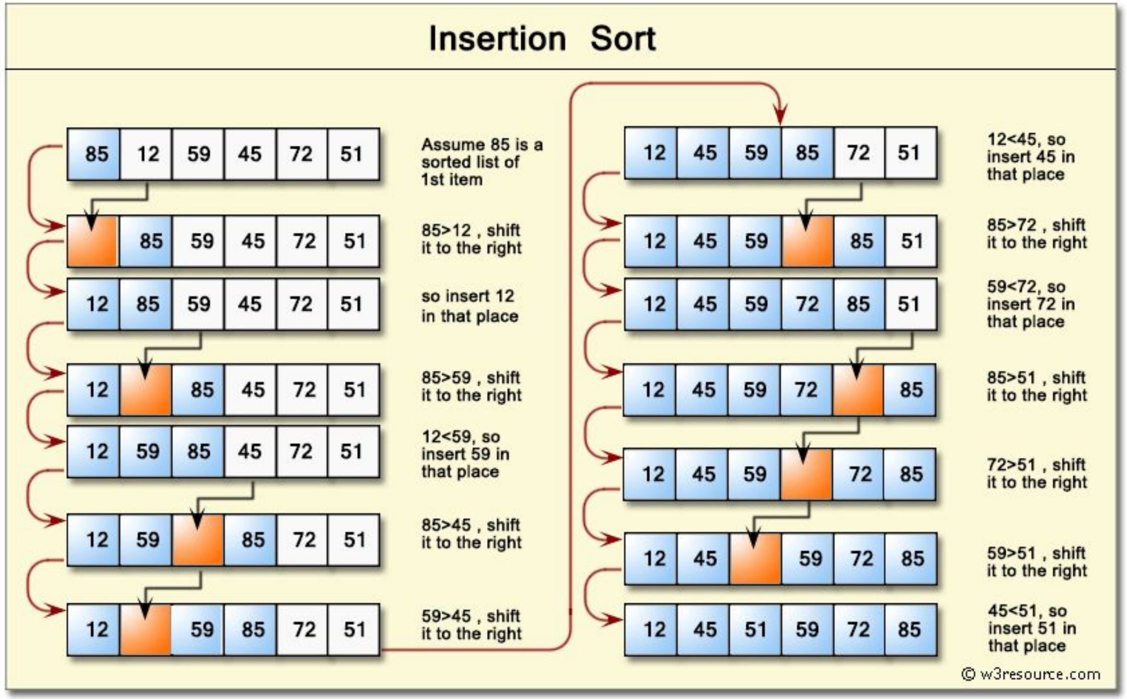


Insertion Sort

- Idea : Similar to how most people arrange a hand of poker cards
  1. Start with one card in your hand
  2. Pick the next card and insert it into its proper sorted order
  3. Repeat previous step for all cards
- Property : Stable



Insertion Sort


Insertion Sort
85 12 59 45 72 51
Assume 85 is a sorted list of 1st item
85 59 45 72 51
85>12 , shift it to the right
12 85 59 45 72 51
so insert 12 in that place
12 85 45 72 51
85>59 , shift it to the right
12 59 85 45 72 51
12<59, so insert 59 in that place
12 59 85 72 51
85>45 , shift it to the right
12 59 85 72 51
59>45 , shift it to the right
12 45 59 85 72 51
12<45, so insert 45 in that place
12 45 59 85 51
85>72 , shift it to the right
12 45 59 72 85 51
59<72, so insert 72 in that place
12 45 59 72 85
85>51 , shift it to the right
12 45 59 72 85
72>51 , shift it to the right
12 45 59 72 85
59>51 , shift it to the right
12 45 51 59 72 85
45<51, so insert 51 in that place
© w3resource.com

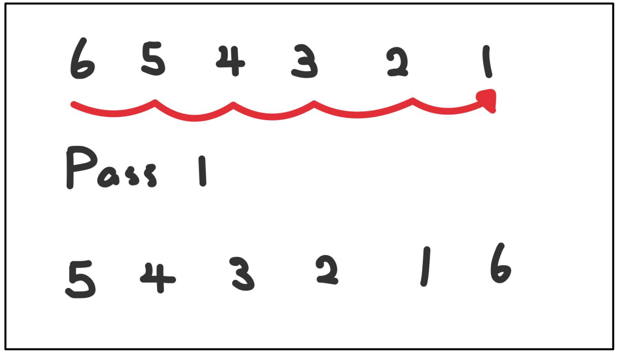
Exercise1

- Implement bubble sort, and you can use the template from e3 (optional).
- Input : unsorted 6 integer numbers
- Output : all pass of the algorithm which sorts those 6 integer numbers (ascending order).

Hint:

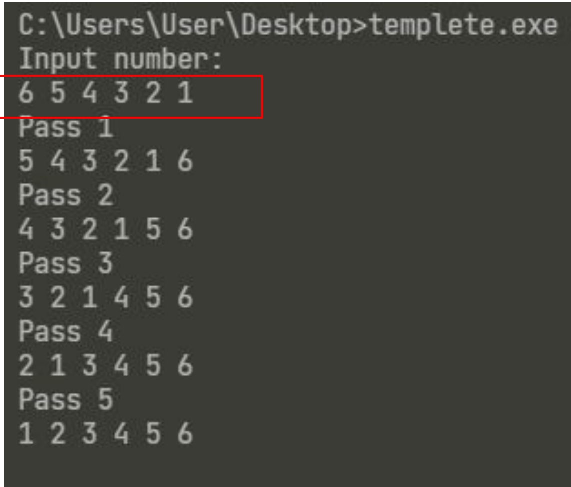
Be careful the process order. (Each pass, put the largest one into right handside)

For example:



Output sample:

Input by user



```
C:\Users\User\Desktop>templete.exe
Input number:
6 5 4 3 2 1
Pass 1
5 4 3 2 1 6
Pass 2
4 3 2 1 5 6
Pass 3
3 2 1 4 5 6
Pass 4
2 1 3 4 5 6
Pass 5
1 2 3 4 5 6
```

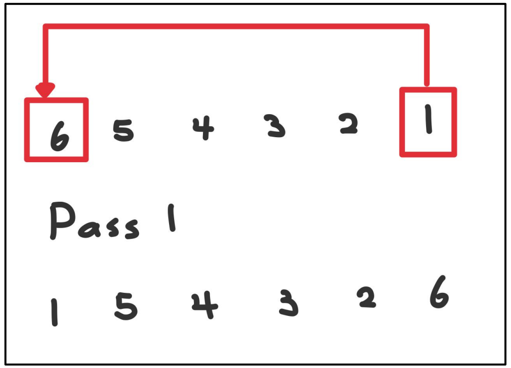
Exercise2

- Implement selection sort, and you can use the template from e3 (optional).
- Input : unsorted 6 integer numbers
- Output : all pass of the algorithm which sorts those 6 integer numbers (ascending order).

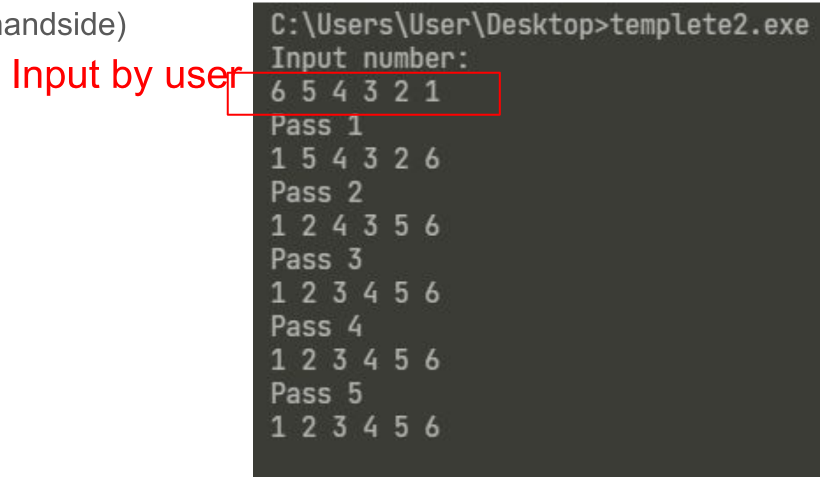
Hint:

Be careful the process order. (Each pass, put the smallest one into left handside)

For example:



Output sample:



```
C:\Users\User\Desktop>templete2.exe
Input number:
6 5 4 3 2 1
Pass 1
1 5 4 3 2 6
Pass 2
1 2 4 3 5 6
Pass 3
1 2 3 4 5 6
Pass 4
1 2 3 4 5 6
Pass 5
1 2 3 4 5 6
```

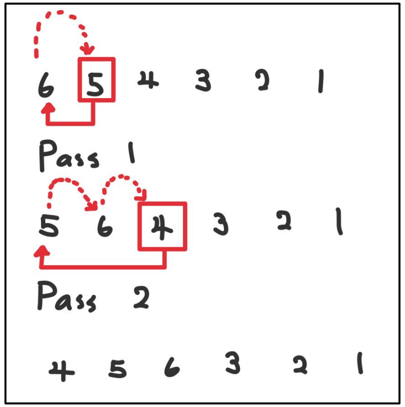
Exercise3

- Implement insertion sort, and you can use the template from e3 (optional).
- Input : unsorted 6 integer numbers
- Output : all pass of the algorithm which sorts those 6 integer numbers (ascending order).

Hint:

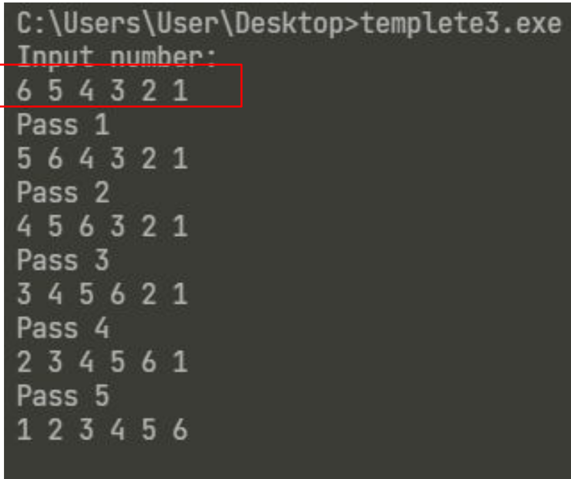
Be careful the process order. (Sort the array from left to right)

For example:



Output sample:

Input by user



```
C:\Users\User\Desktop>templete3.exe
Input number:
6 5 4 3 2 1
Pass 1
5 6 4 3 2 1
Pass 2
4 5 6 3 2 1
Pass 3
3 4 5 6 2 1
Pass 4
2 3 4 5 6 1
Pass 5
1 2 3 4 5 6
```

Exercise Submission Format

Format:

- xxxxxxxxx_ex_w14.zip
    - xxxxxxxxx_ex_01.cpp
    - xxxxxxxxx_ex_02.cpp
    - xxxxxxxxx_ex_03.cpp

xxxxxxxxx is your student ID