Data Structures
Homework Solution 7 and 8

By Po-Chuan & Pei-Hsuan

HW 7

7.1

Write a C++ function that, given as input two objects of a class that implements the ADT list, returns a new list obtained from concatenating the two lists (by appending the second to the end of the first). The function should behave like a client of the ADT list, independent from its implementation

7.1

```
template <typename ItemType>
List<ItemType>::concatenate_list(List<ItemType> A_list, List<ItemType> B_list)
{
	List<Itemtype> newList = new List<ItemType>();
	int alen = A_list.getLength();
	int blen = B_list.getLength();

	for (int i=1; i<alen; i++) newList.insert(i, a.getEntry(i));
	for (int i=1; i<blen; i++) newList.insert(i+alen, b.getEntry(i));
	return newList;
}
```

7.1 Grading Policy

- function prototype (2 lists) 4
- return a new list 4
- appends correctly 8
- client function 4

7.2 Exercise 10.5

Consider the following C++ function f, which calls the function swap. Assume that swap exists an simply swaps the contents of its two arguments. How many comparisons does f perform? Count only the number of comparisons between two entries of the array.

```
void f(int theArray[], int n)
{
        for (int j = 0; j < n; ++j)
        {
                int i = 0;
                while (i <= j)
                {
                        if (theArray[i] < theArray[j])
                                swap(theArray[i] , theArray[j]);
                        i++;
                }
        }
}
```

j=0, i will run 1 time
j=1, i will run 2 times
j=2, i will run 3 times
j=3, i will run 4 times
…
j=n-1, i will run n times

So, total is

$$1 + 2 + 3 + \cdots + n = \frac{n(n+1)}{2}$$

7.2 Grading Policy

- correct answer 20

7.3 Exercise 10.9

Show that $7n^2 + 5n$ is not $O(n)$

1. According to the definition of order O(), algorithm A is order f(n) – denoted O(f(n)) – if constants k and n_0 exist such that A requires no more than k × f(n) time units to solve a problem of size $n \geq n_0$.
2. Assume there exists a k and for $n \geq n_0$ that $7n^2 + 5n \leq cn$.
3. However, we can't find a k for $n \geq n_0$ to match this formula.
4. Therefore, $7n^2 + 5n$ is not $O(n)$.

7.4 Grading Policy

- smooth explanation or proof 20
- deductions for errors

7.4 Exercise 11.2

Trace the insertion sort as the following array into ascending order: 45 13 27 52 18 25 Show which pair of entries are compared and which pair are exchanged as the algorithm executes.

7.4 Exercise 11.2

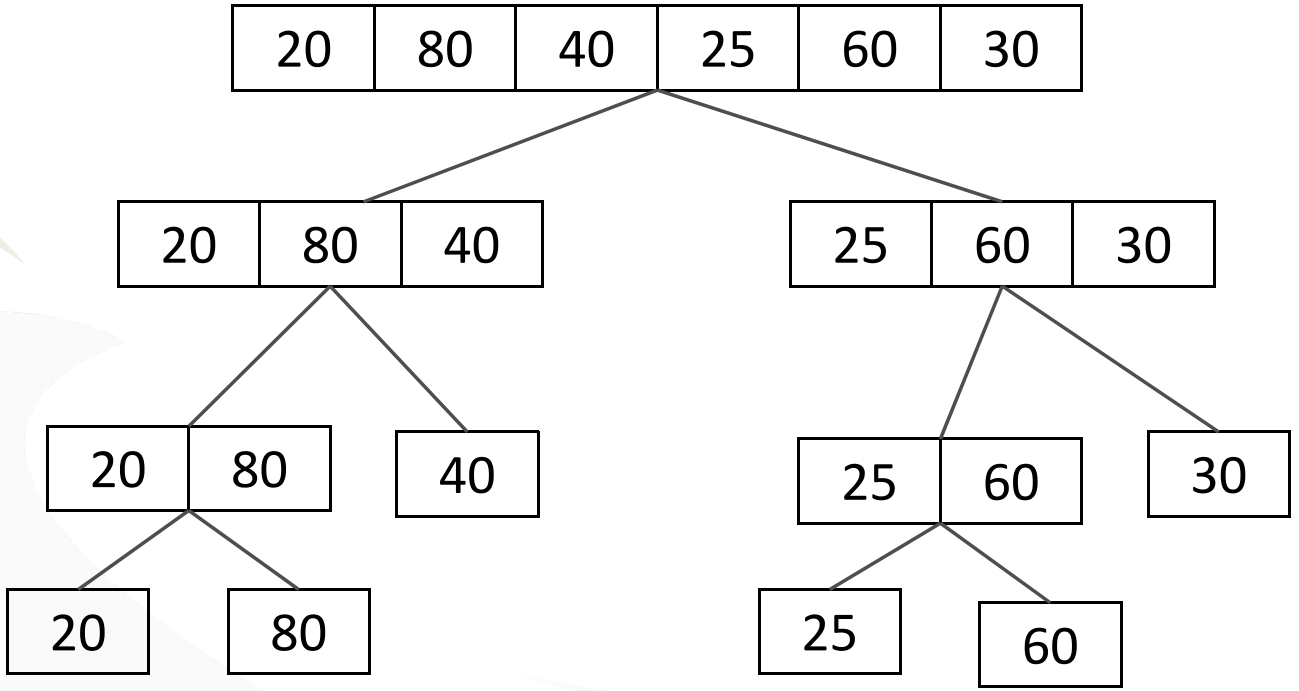
Index	[1]	[2]	[3]	[4]	[5]	[6]	
1	45	13	27	52	18	25	Copy 13
2	13	45	27	52	18	25	Shift 45
3	13	27	45	52	18	25	Insert 13, copy 27
4	13	27	45	52	18	25	Shift 45
3	13	27	45	18	52	25	Insert27, copy 52, insert 52
2	13	27	18	45	52	25	Copy 18
4	13	18	27	45	52	25	Shift 27, 45, 52
5	13	18	27	45	52	25	Insert 18, copy 52
4	13	18	27	45	25	51	Shift 27, 45, 52
5	13	18	27	25	45	51	Insert 25
	13	**18**	**25**	**27**	**45**	**51**	

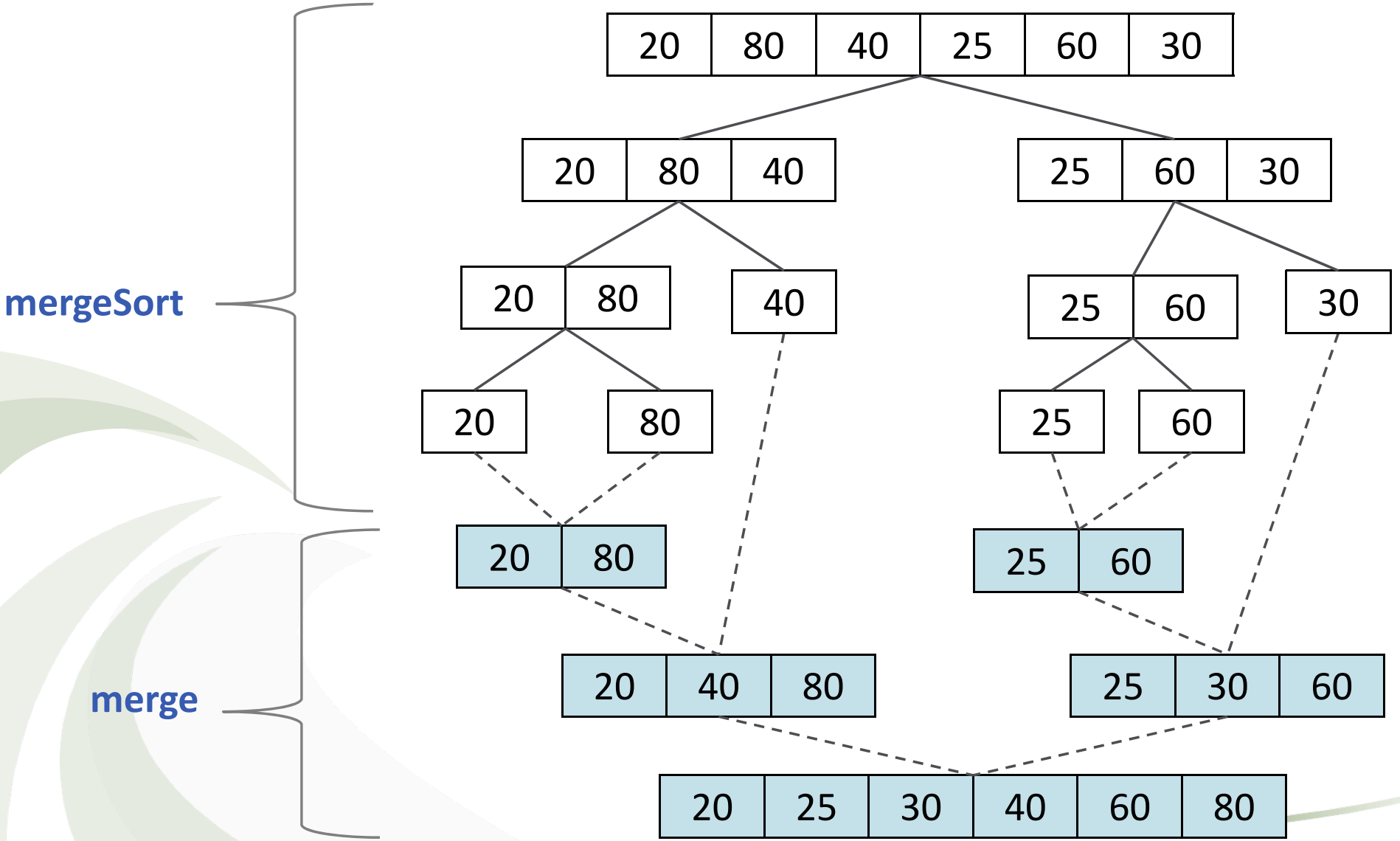
7.4 Grading Policy

- each stage 4
- each error -1
- Trace another sort method -8

7.5 Exercise 11.10

Trace the merge sort algorithm as it sorts the following array into ascending order. List the calls to mergeSort and merge in the order in which they occur. 20 80 40 25 60 30




mergeSort
merge
20 80 40 25 60 30
20 80 40
25 60 30
20 80
40
25 60
30
20
80
25
60
20 80
25 60
20 40 80
25 30 60
20 25 30 40 60 80

7.5 Grading Policy

- each divide and conquer 3
- wrong division or merge -1
- Didn’t list down mergeSort(), merge() -5

HW 8

8.1 Exercise 13.7

(You should provide an appropriate declaration and also an implementation for the new display operation, both in C++.)

An operation that displays the contents of a queue can be useful during program debugging. Add a display operation to the ADT queue such that display uses only ADT queue operation, so it is independent of the queue's implementation.

8.1 Exercise 13.7

```
template<class ItemType>
bool implementQueue<ItemType>::display_queue(const implementQueue& aQueue)
{
	while( !aQueue.isEmpty() )
	{
		cout << aQueue.peekFront() << endl;
		aQueue.dequeue();
	}
	return true;
}
```

`implementQueue` can be `ListQueue` or `ArrayQueue`

8.2 Exercise 13.11

(Note that the customer being served is not part of the queue representing the waiting line.)

With the following data, hand-trace the execution of the bank-line simulation that this chapter describes. Each line of data contains an arrival time and a transaction time. Show the state of the queue and the event list at each step.

5 9

7 5

14 5

30 5

32 5

34 5

Note that at time 14, there is a tie between the execution of an arrival event and a departure event.

By using the arrival time and the transaction length, the simulation can easily determine the time at which a customer departs.

Data

5 9
7 5
14 5
30 5
32 5
34 5

	Arrive	**length**	**Start**	**Departure**
Customer1	5	9	5	14
Customer2	7	5	14	19
Customer3	14	5	19	24
Customer4	30	5	30	35
Customer5	32	5	35	40
Customer6	34	5	40	45

Use a **queue** to represent the line of customers in the bank. Let's trace the bank simulation algorithm.

Time	bankQueue	eventList_PrioirtyQueue
0		A 5 9 | A 7 5 | A 14 5 | A 30 5 | A 32 5 | A 34 5
5		A 7 5 | A 14 5 | D 14 - | A 30 5 | A 32 5 | A 34 5
7	A 7 5	A 14 5 | D 14 - | A 30 5 | A 32 5 | A 34 5
14	A 14 5	D 14 - | D 19 - | A 30 5 | A 32 5 | A 34 5
19	A 14 5	D 24 - | A 30 5 | A 32 5 | A 34 5
24		A 30 5 | A 32 5 | A 34 5

Time	bankQueue	eventList_PrioirtyQueue
24		A 30 5 | A 32 5 | A 34 5
30		A 32 5 | A 34 5 | D 35 -
32	A 32 5	A 34 5 | D 35 -
34	A 32 5 | A 34 5	D 35 -
35	A 34 5	D 40 -
40		D 45 -

8.3 Exercise 14.2 (Use C++)

Repeat the previous exercise, but implement a memory-safe copy constructor instead. If a memory allocation fails, this constructor should release all memory that was allocated prior to the failure and then throw an exception.

Previous exercise: Implement the copy constructor for the class LinkedQueue that is declared in Listing 14-3. Hint: look at the copy constructor for the ADT stack in Listing 7-4 of Chapter 7.

8.3 Exercise 14.2 (Use C++)

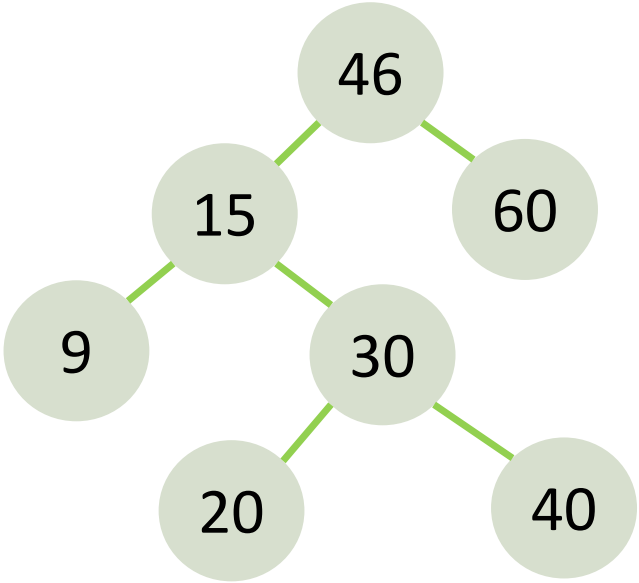
```
template<class ItemType>
ListQueue<ItemType>::ListQueue( const ListQueue& aQueue )
{
	for ( auto i = aQueue.listPtr; i != nullptr; i = i->getNext() )
		if ( ! enqueue( i->getItem() ) )
		{
			delete this;
			throw PrecondViolatedExcep( "Memory allocation failed." );
		}
}
```

8.4 Exercise 14.2

Using the binary search tree, write the sequence of nodes visited in

a. preorder b. postorder c. inorder

preorder (root → left → right)	46 → 15 → 9 → 30 → 20 → 40 → 60
postorder (left → right → root)	9 → 20 → 40 → 30 → 15 → 60 → 46
inorder (left → root → right)	9 → 15 → 20 → 30 → 40 → 46 → 60



8.5 Exercise 15.11

Consider the binary search tree in Figure15-18. What tree results after you insert the nodes in that order. Draw the tree resulted from inserting 90, 25, 65, 27, 57, and 10 or (90, 25,45,27,57 and 10) in that order and also the tree from inserting the same nodes but in the reversed order. Show just the final result; no need to draw the intermediate trees.