OOP11 MACHINE TEST

General information

1 Time 2011/6/16 6:30~10:30 pm

2 Score 4 problems; 100 points + 10 bonus points

3 Files

Problem	Score	Files
A	20	A.cpp, slist.h
B	20 (+10 bonus points)	B.cpp
C	24	C.cpp, vector7.h
D	36	D.cpp, list8.h

4 General requirements

- Open course web site, lecture notes, homework solutions, and nothing else.
- Use the stipulated algorithm, if any.
- Comments are not required.
- Suffice it to run the sample test. However, you shall present a general solution for each problem – any solution tailored for the sample test data will come to nought.
- Be honorable!
  Any activity unrelated to the test such as browsing the web, chatting on web, playing game, etc., is strictly prohibited.

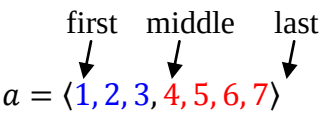
Problem A

Generic algorithm

Write the following generic algorithm to exchange the two subsequences **`[first, middle)`** and **`[middle,last)`** using forward iterators.

```
template<class ForwardIterator>
void rotate(ForwardIterator first,ForwardIterator middle,
                                  ForwardIterator last);
```

For example, consider



By exchanging the blue-colored and red-colored subsequences (or, equivalently, by rotating the sequence 3 places to the left), we obtain
$a = \langle 4, 5, 6, 7, 1, 2, 3 \rangle$

Requirement

Your algorithm shall run in-place, i.e. use only $O(1)$ extra space. In other words, it is forbidden to use an auxiliary sequence.

Sample test

Suffice it to run the sample test contained in file A.cpp.
For testing purpose, the file "slist.h" is included in A.cpp and contains an extension to the example on singly-linked list given in lecture.

Sample run

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

Problem B

Dynamic programming: Set partition

Recall the set partition problem of HW#4 and HW#5.

Given a set $A = \{a_1, a_2, \dots, a_n\}$ of positive integers, count the number of subsets $A' \subset A$ such that

$$\sum_{a \in A'} a = \sum_{a \in A - A'} a \qquad (*)$$

Let $b = \sum_{a_i \in A} a_i$

If b is odd, then no such subset exists.

If b is even, this problem may be solved as follows.

For $1 \le i \le n,\ 0 \le j \le b/2$, define

$t[i,j] =$ the number of subsets of $\{a_1, a_2, \dots, a_i\}$ for each of which the sum of the elements is exactly j

by

$t[1,j] = 1$, if $j = 0$ (i.e. the subset is $\emptyset$)
or $j = a_1$ (i.e. the subset is $\{a_1\}$)
$= 0$, otherwise (i.e. $j > 0$ but $j \ne a_1$)

$t[i,j] = t[i-1, j-a_i] + t[i-1, j]$, if $i > 1$ and $j \ge a_i$
$= t[i-1, j]$, if $i > 1$ and $j < a_i$

The value we want is $t[n, b/2]$.

Requirement

You shall solve this problem by the technique of dynamic programming. That is to say, you shall

(1) use STL **`vector`** class to create an $n \times (1 + b/2)$ table t, and

(2) fills in the table t *iteratively*

For testing purpose, you shall write the following function

```
int subset(int* a,int n);
```

that returns the number of subsets satisfying equation (*).

Bonus (10%)

You will earn 10 bonus points if you also show ***any*** subset satisfying equation (*), in case the partition is possible. Again, you shall use dynamic programming technique. To this end, you shall use STL **`vector`** class to create an $n \times (1 + b/2)$ table s, and use it to record a subset satisfying equation (*), if possible.

Sample test

Suffice it to run the sample test contained in file B.cpp.

Sample run

```
Test 1...
8 subset(s) in total

Test 2...
14 subset(s) in total

Test 3...
0 subset(s) in total
```

Sample run (for bonus)

```
Test 1...
One subset: 2 3 4 5
8 subset(s) in total

Test 2...
One subset: 1 2 4 5 6
14 subset(s) in total

Test 3...
0 subset(s) in total
```

Problem C

STL Vector

Extend the **`vector`** class of HW#7 with the member function

```
vector<T>& operator+=(const vector<T>& rhs);
```

This function appends the vector referenced to by **`rhs`** to the end of the vector pointed to by **`this`** and returns a reference to the vector pointed to by **`this`**.

For example, let

$a = (a_1, a_2, \cdots, a_m)$

$b = (b_1, b_2, \cdots, b_n)$

be two vectors, where $m = a.\text{size}()$ and $n = b.\text{size}()$, then

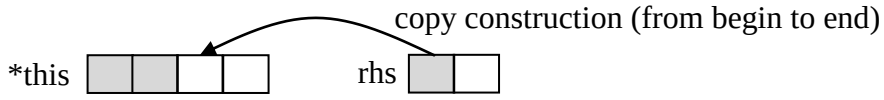
$a \mathrel{+}= b$

sets vector a to $(a_1, a_2, \cdots, a_m, b_1, b_2, \cdots, b_n)$ and returns a reference to vector a. The vector b remains unchanged.

Storage allocation requirement

Case 1: $a.\text{capacity}() \geq m + n$

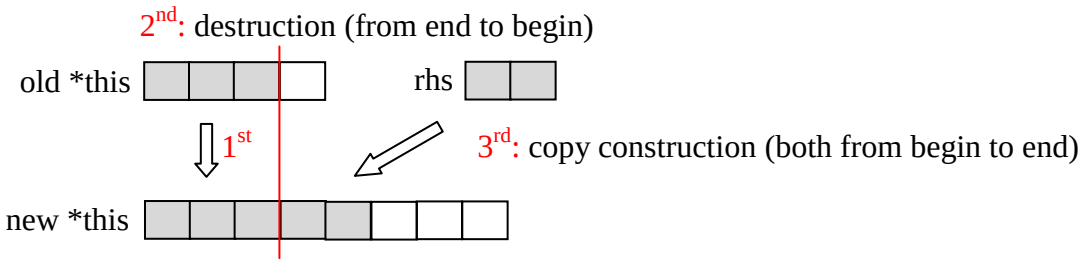
Leave the capacity of vector a unchanged.



Case 2: $a.\text{capacity}() < m + n$

Set the new capacity of vector a to the least power of 2 that is greater than or equal to $m + n$.

For example, if $m + n = 4$, set the new capacity of vector a to 4; and if $m + n = 5$, set the new capacity of vector a to 8.



Hint

DO NOT use **`insert`** or **`push_back`**, as they won't meet the requirement.

Sample test

Suffice it to run the sample test contained in file C.cpp.

`operator+=` has already been declared within the **`vector`** class defined in file vector7.h.

You need only define **`operator+=`** outside the class body.

Sample run

```
Test 1...
1 2 1 2
4
1 2 1 2 1 2 3 4 5
16
1 2 1 2 1 2 3 4 5 1 2 3 4 5
16
1 2 1 2 1 2 3 4 5 1 2 3 4 5 1 2 1 2 1 2 3 4 5 1 2 3 4 5
32

Test 2...
mermaids copy-constructed

Test 3...
doraemon copy-constructed
garfield copy-constructed
mermaids copy-constructed
mermaids destructed
garfield destructed
doraemon destructed
doraemon copy-constructed
garfield copy-constructed
mermaids copy-constructed
```

Problem D

List operations

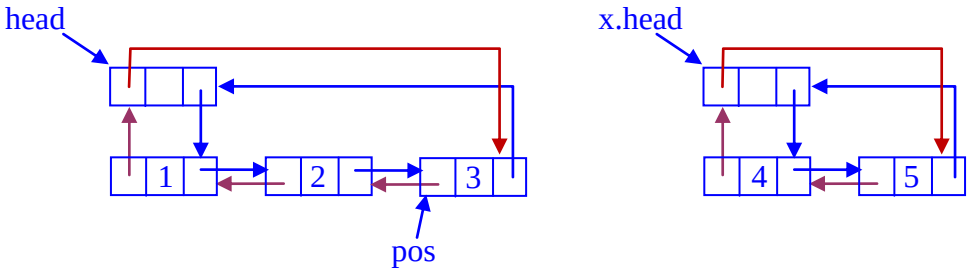
Part D1

Extend the **`list`** class of HW#8 with the member function

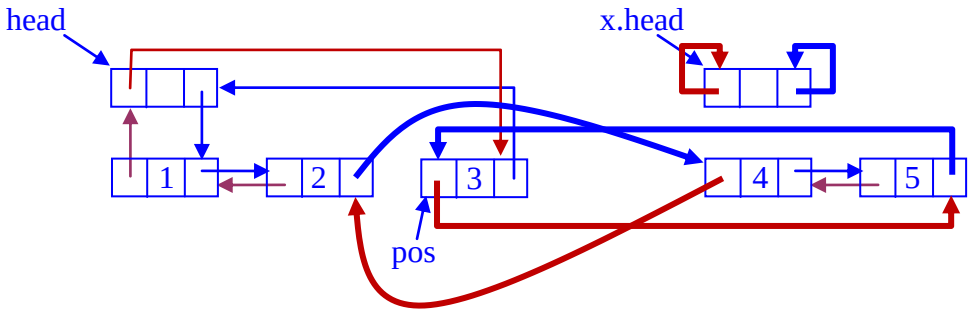
```
void splice(iterator pos,list<T>& x);
```

Assuming that **`this!=&x`**, this function inserts the contents of **`x`** before **`pos`** and empties **`x`**, as follows.

Before:



After:



Hint:

Adjust the 3 heavy blue-colored pointers and the 3 heavy brown-colored pointers.

As in HW#8, there are two ways for **`list::splice()`** to access the private member of the **`iterator`** class. ***Choose one!***

Sample test

Suffice it to run the sample test contained in file D.cpp.

`splice` has already been declared within the **`list`** class defined in file list8.h.

You need only define **`splice`** outside the class body.

Part D2

Extend the **list** class of HW#8 to enable the code in line 3 below.

```
1   char a[]="s@n#o!o$p%y";
2   list<char> b(a,a+sizeof(a)-1);
3   list<int> c(b);
```

Line	Comment
2	Initialize list **b** with the sequence of characters ⟨s, @, n, #, o, !, o, $, p, %, y⟩
3	This line uses the character list **b** to construct the int list **c** as follows: a) copy the 1st, 3rd, 5th, … characters of **b** to **c** (the characters are converted to integers) b) retain the 1st, 3rd, 5th, … characters of **b** c) remove the 2nd, 4th, 6th, … characters from **b** After the destructive and converting construction, we have **b** = ⟨s, n, o, o, p, y⟩ **c** = ⟨115, 110, 111, 111, 112, 121⟩ where the integers contained in **c** are the ASCII codes of the corresponding characters contained in **b**.

Hint

To enable the code in line 3, you have to define a templated, destructive converting ctor that does the conversion **list<U>** → **list<T>**, assuming that the conversion **U** → **T** exists.

Sample run

Suffice it to run the sample test contained in file D.cpp.
Be sure to add the following two things to the file list8.h.
(1) Declare the templated, destructive converting ctor within the class body
(2) Define it outside the class body

Sample run for both parts

```
Testing part D1 ...
List b: 1 2 3 4 5 1 2 3 4 5 6 7 8 9
List c:
List b: 1 2 3 4 5 1 2 3 4 5 6 7 8 9 6 7 8 9

Testing part D2 ...
s n o o p y
115 110 111 111 112 121
s o p
115 111 112
```