W.P.Hwang
CS-CCS-NCTU

MACHINE TEST

INTRO. TO COMPUTERS & PROGRAMMING

FALL 2009

General information

1 Time 2010/1/5 6:30~10:30 pm

2 Score 4 problems; A (20%), B (25%), C (25%), D (30%)

3 General requirements

- Use the stipulated algorithm, if any.
- Comments are not required.
- Each problem has a downloadable sample test and a sample output. (You may also download all the sample tests here.) 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.

4 Source file name

Problem A <PID>A.cpp Note: <PID> is your personal identifier.
Problem B <PID>B.cpp
Problem C <PID>C.cpp
Problem D <PID>D.cpp

Also, write down your name and student ID in the first line of each source file as a comment, i.e.

// <name> <student ID>

Problem A

Common digits

Write the following function

```
int common(int m,int n);
```

to count the number of distinct digits that occur in both **m** and **n**, e.g.

m	n	common(m,n)	digits-in-common
123454321	13575	3	1,3,5
56789	1234	0	none
2010	0	1	0
-56789	-12345	1	5
-2147483648	2147483647	7	1,2,3,4,6,7,8

Only the number of distinct digits is required; the distinct digits themselves needn't be displayed.

Sample test

Click here for the file that contains a sample test for this problem.

Sample output

```
3
0
1
1
7
```

Problem B

Subset generation

Given an integer $n \geq 0$, count and generate the subsets of the set $\{1, 2, 3, \ldots, n\}$.

Requirements

1 Assume that $n \leq 10$

2 Solve this problem in a manner similar to the recursive algorithm for the k-combinations problem given in the lecture.

Let

$p(n)$ = the number of subsets of the set $\{1, 2, 3, \ldots, n\}$

Then,

$$p(n) = 1, \quad \text{if } n = 0$$
$$= p(n-1) + p(n-1), \quad \text{otherwise}$$

the number of subsets that contain n

the number of subsets that don't contain n

3 See sample run for the required output format.

Sample test

Click here for the file that contains a sample test for this problem.
To make the sample test runnable, you have to define the function

```
int p(int n);
```

to count and display the subsets of $\{1, 2, \ldots, \mathbf{n}\}$.

(*Continued on the next page*)

Sample output

```
Enter an integer <= 10: 3
{}
{1}
{2}
{1,2}
{3}
{1,3}
{2,3}
{1,2,3}
There are 8 subsets.

Enter an integer <= 10: 4
{}
{1}
{2}
{1,2}
{3}
{1,3}
{2,3}
{1,2,3}
{4}
{1,4}
{2,4}
{1,2,4}
{3,4}
{1,3,4}
{2,3,4}
{1,2,3,4}
There are 16 subsets.

Enter an integer <= 10:
```

Problem C

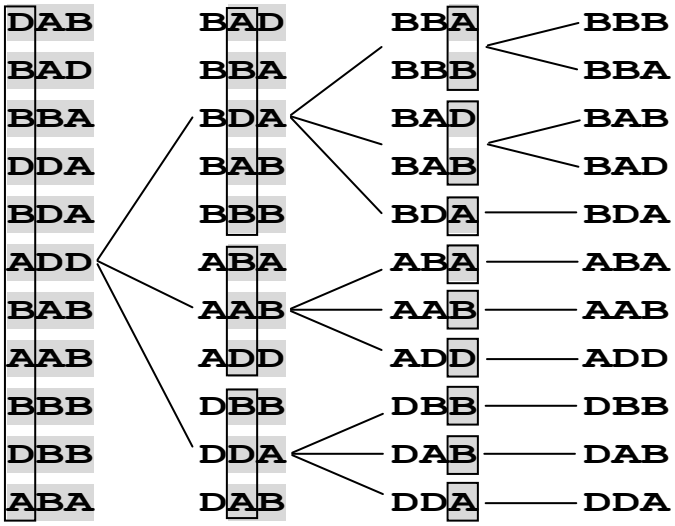
BAD quicksort

A BAD number is a hexadecimal integer that contains only hexadecimal digits B, A, and D.

A 4-byte unsigned integer may be used to store a *k*-digit BAD number, for $1 \le k \le 8$. For example, the following unsigned array contains 3-digit BAD numbers:

```
unsigned a[11]={0xDAB,0xBAD,0xBBA,0xDDA,0xBDA,0xADD,
                0xBAB,0xAAB,0xBBB,0xDBB,0xABA};
```

Define the order: B < A < D
With this ordering, the preceding array of 3-digit BAD numbers may be sorted into nondecreasing order in a manner similar to the binary quicksort of HW8.



For this problem, you are asked to use the aforementioned algorithm to sort an unsigned array of *k*-digit BAD numbers, $1 \le k \le 8$.

(*Continued on the next page*)

Sample test

Click here for the file that contains a sample test for this problem.
To make the sample test runnable, you have to define the function

```
void BAD_qsort(unsigned a[],int n,int k);
```

to sort an unsigned array **a** of **n** BAD numbers, each having **k** digits.
Hint: As in HW8, you have to define another function that is invoked by **BAD_qsort** to actually sort the array.

Sample output

```
Test 1 ...
BBB BBA BAB BAD BDA ABA AAB ADD DBB DAB DDA
Test 2 ...
BBB BBA BBD BAB BAA BAD BDB BDA BDD ABB ABA ABD AAB AAA AAD ADB ADA ADD DBB DBA
DBD DAB DAA DAD DDB DDA DDD
Test 3 ...
BBBBBB BBBAAA BBBDDD BBDDAA AABBDD AAABBB AAAAAA AAADDD DDAABB DDDBBB DDDDDD
```

Problem D

Sorted singly-linked lists

In this problem, you are asked to maintain a sorted singly-linked list. More precisely, your job is to modify the lecture example on singly-linked lists as follows:

Command	Action
`i` *n*	insert the integer *n* to the already-sorted list while keeping the resulting list sorted
`d` *k*	if *k*=0, erase all nodes in the list that contain even integers, if any if *k*=1, erase all nodes in the list that contain odd integers, if any (You may assume that there is no other value of *k*.)

See the sample run below to confirm the behaviors of these two commands.

Requirements

1 Among the two operations **insert** and **erase**, it is required that one of them be coded iteratively, and the other recursively. To this end, your score of this problem will depend on how the operations are coded.

Score	Coding method
24%	Both use recursion.
24%	Both use iteration.
30%	One uses recursion and the other uses iteration (Which uses recursion is up to you.)

2 The two operations **insert** and **erase** will be graded independently. In particular, if your own insert function doesn't work, feel free to use the original insert function from the lecture example (given in the sample test) to test your erase function.

Sample test

Click here for the file that contains the lecture example tailored for this problem. In particular, the responses to commands `i` and `d` within the toy user interface are omitted and left to you.

Sample output

```
Command: i5
Command: i2
Command: i8
Command: i3
Command: i9
Command: i1
Command: i6
Command: p
1 2 3 5 6 8 9
Command: d0
Command: p
1 3 5 9
Command: i6
Command: i4
Command: i7
Command: p
1 3 4 5 6 7 9
Command: d1
Command: p
4 6
Command: d1
Command: p
4 6
Command: d0
Command: p

Command:
```