CCU CSIE
Data Structure Course – 2016 Fall
Computer-Based Test I

Teacher: Yu-Ling Hsueh
TA: Mike, Wayne, Jay

November 18th, 2016

Notes:

- 6 problems in total. 25% for each. 100% at most.
- No talking, no cheating.
- Leave your bag, smart phone, and books in the front of the classroom.
- Internet has been cut off, so just ask TAs when you have any questions.

1 Invert a singly linked list.

Please implement a function to **invert a singly linked list** in C. The input format is shown as Input1 and Input2.

Note 1: You must use a linked list; otherwise, no points will be given.

Test Case

Please test your program with Input1 and Input2, and then check the answers with Output1 and Output2.

Listing 1: Invert a Singly Linked List

```
Input1:
1 4 5 0 6 7 2

Output1:
2 7 6 0 5 4 1

Input2:
9 6 13 26 78 32 65

Output2:
65 32 78 26 13 6 9
```

2 Infix to Postfix Conversion

Please finish a program which can change the expression from the infix form to the postfix form in C **using a stack**. The input operands are lowercase letters $\lceil a \rfloor$ to $\lceil z \rfloor$; meanwhile, the input operators are $\lceil * \rfloor$, $\lceil / \rfloor$, $\lceil + \rfloor$, and $\lceil - \rfloor$. Furthermore, the precedence of the operators is $\lceil * \rfloor = \lceil / \rfloor > \lceil + \rfloor = \lceil - \rfloor$. The output includes two lines of the results: (1) the Postfix expression, and (2) the maximum number of elements in the stack during the computation.

Test Case

Please test your program with Input and Input2, and then check the answers with Output1 and Output2.

Listing 2: Infix to Postfix Conversion

```
Input1:
a+b*c

Output1:
abc*+
2

Input2:
a/b-c+d*e-a*c

Output2:
ab/c-de*+ac*-
2
```

3 Binary Search Tree

Implement a binary search tree with a delete(x) function **using an array**. Print out the Pre-order traversal after the delete function is operated. The input consists of a sequence of numbers in the first line. The delete function and the number to be deleted are shown in the second line. You must build a binary search tree according to the input order. The output includes three lines of the results: (1) the contents of the array that stores the binary search tree, (2) the contents of the array that stores the binary search tree after the delete operation is operated, and (3) the Pre-order in (2).

Note 1: If the values are null, replace the null by x.
Note 2: For the delete function, find the replacement from the left sub-tree first, if the left sub-tree is not empty. Otherwise, find the replacement from the right sub-tree.

Test Case

Listing 3: Binary Search Tree

```
Input:
4 5 1 0 6 7 2
delete(5)

Output:
4 1 5 0 2 x 6 x x x x x x x 7
4 1 6 0 2 x 7
4 1 0 2 6 7
```

4 Heap Sort

Given a sequence of n numbers, build a heap **using an array** and implement a heap sort that results in a sorted numbers in ascending order. The output includes two lines of the results: (1) the contents of the array that stores the heap, and (2) the sorted numbers in ascending order.

Test Case

Please test your program with Input1 and Input2, and then check the answers with Output1 and Output2.

Listing 4: Heap Construction and Heap sort

```
Input1:
1 4 5 0 6 7 2

Output1:
0 1 2 4 6 7 5
0 1 2 4 5 6 7

Input2
3 7 5 9 1

Output2:
1 3 5 9 7
1 3 5 7 9
```

5 Equivalent Relations

Given n equivalent relations, find the equivalent classes. Each line in the input shows the equivalent relation that consists of a pair of characters separated by a space. Each line in the output is an equivalent class.

Note 1: You must sort each equivalent class in ascending order.

Test Case

Please test your program with Input1 and Input2, and then check the answers with Output1 and Output2.

Listing 5: Equivalent Relations

```
Input1:
0 4
3 1
6 10
8 9
7 4
6 8
3 5
2 11
11 0
```

```
11
12 Output1:
13 0 2 4 7 11
14 1 3 5
15 6 8 9 10
16
17 Input2:
18 1 3
19 2 5
20 1 7
21 5 6
22 6 0
23
24 Output2:
25 1 3 7
26 0 2 5 6
```

6 Maze

Given a 10 by 10 maze with one entrance and one exit, find the way to the exit (8,8) from the entrance (1,1). There is a border (represented by 1s) around the maze. There are only four moving directions: right > down > left > up. You must use a stack to store the moving path. Input is a 10 by 10 maze with a border. Output should be the moving path that consists of steps separate by ",".

Test Case

Listing 6: Maze

```
1  Input:
2  1111111111
3  1001000111
4  1101010001
5  1100010101
6  1101110101
7  1001110111
8  1011000111
9  1011011111
10 1011000001
11 1111111111
```

Output

(1,1),(2,1),(2,2),(2,3),(3,3),(4,3),(4,2),(4,1),(5,1),(6,1),(6,2),(7,2),(8,2),(8,3),(8,4),(8,3),(8,2),(7,2),(6,2),(6,3),(6,4),(6,5),(6,6),(5,6),(4,6),(4,7),(4,8),(5,8),(6,8),(7,8),(8,8)