1. (3pts) Which is the least essential operation to define for (pure) stack data type?
   甲. boolean empty() - Tests if this stack is empty
   乙. Object peek() - Looks at the object at the top of this stack without removing it from the stack.
   丙. void push(Object item) - Pushes an item onto the top of this stack.
   丁. Object pop() - Removes the item at the top of this stack and returns that item as the value of this function.
   戊. size() - Returns the number of items in this stack.

2. (3pts) An implementation of the stack data type is sketched below:
   ```java
   Object[] items = new Object[1];
   int top = -1;
   
   boolean empty() { return top == -1; }
   void push(Object obj) {
     if (top == items.length - 1) {
       enlarge();
     }
     items[++top] = obj;
   }
   private void enlarge() {
     Object[] tmp = new Object[items.length * 2];
     for (int i = 0; i < items.length; i++) {
       tmp[i] = items[i];
     }
     items = tmp;
   }
   Object pop() {
     Object obj = items[top];
     items[top--] = null;
     return obj;
   }
   Object peek() { return items[top]; }
   ```
   What is the worst case time complexity best characterizing the push() operation, assuming n is the total number of operations applied to the stack.
   甲. O(1)
   乙. O(log n)
   丙. O(n)
   丁. O(n log n)

3. (3pts) As in Question 3, what is the worst case time complexity best characterizing pop()?
   甲. O(1)
   乙. O(log n)
   丙. O(n)
   丁. O(n log n)

4. (3pts) As in Question 3, what is the amortized time complexity best characterizing push()?
   甲. O(1)
   乙. O(log n)
   丙. O(n)
   丁. O(n log n)
5. (3pts) The figure below shows a maze where A is the entrance, Y the exit, and gray cells the walls. Assume one can move in four directions (north, south, east, and west). To search for a feasible path from A to Y, it is natural to use a stack to record visited paths while in the middle of the maze.

```
A B C D E
F G H I J
K L M N O
P Q R S T
U V W X Y
```

Assume that after entering a cell its neighboring, not-yet-visited cells are tried in alphabetical order. What is the least likely configuration of the stack during the search (assuming stack grows to the right)?

甲. ABGLMRW
乙. ABGLMNOJ
丙. ABGLKPU
丁. ABCD

6. (3pts) Which is the infix representation of the postfix expression abc+d*e/f-g+,
assuming normal operator priority?

甲. \((a - (b + c) \times d / e - f) + g\)
乙. \((a + (b + c) \times (d / e) - f + g\)
丙. \((a - ((b + c) \times (d / e)) - f + g\)
丁. \((a - (b + c) \times (d / e)) - f + g\)

7. (3pts) Which code segment below implements the removal operation correctly for singly linked lists?

甲. ```java
ListItem remove (ListItem item) {
    ListItem next = item.next;
    item.next = null;
    return next;
}
```  
乙. ```java
ListItem remove (ListItem item) {
    item.next = item.next.next;
    return item.next;
}
```  
丙. ```java
ListItem remove (ListItem item) {
    if (item.next == null) return null;
    ListItem next = item.next;
    item.next = next.next;
    return item.next;
}
```  
丁. ```java
ListItem remove (ListItem item) {
    if (item == null || item.next == null) return null;
    ListItem next = item.next;
    item.next = next.next;
    return next;
}
```
8. (3pts) Which code segment below implements the insertion operation correctly for doubly linked circular lists?

甲. `void insert (DLIST Item anchor, DLIST Item item) {
   item.right = anchor;
   item.left = anchor.left;
   anchor.left.left = item;
   anchor.left = item;
}

乙. `void insert (DLIST Item anchor, DLIST Item item) {
   item.left = anchor;
   item.right = anchor.right;
   anchor.right.right = item;
   anchor.right = item;
}

丙. `void insert (DLIST Item anchor, DLIST Item item) {
   item.left = anchor;
   item.right = anchor.right;
   anchor.right.left = item;
   anchor.right = item;
}

丁. `void insert (DLIST Item anchor, DLIST Item item) {
   item.right = anchor;
   item.left = anchor.left;
   anchor.left.right = item;
   anchor.left = item;
}

9. (3pts) Following the conventions of LISP, assume nodes with two data members: HEAD and TAIL. If A = ((a(bc))), then HEAD(A) = (a(bc)), TAIL(A) = NIL,
HEAD(HEAD(A)) = a, and TAIL(HEAD(A)) = ((bc)). CONS(A, B) gets a new node T, stores A in its HEAD, B in its TAIL, and returns T. B must always be a list.
If L = a and M = (bc), then CONS(L, M) = (abc) and CONS(M, M) = ((bc)bc).
Three other useful functions are: ATOM(X) which is true if X is an atom else false,
NULL(X) which is true if X is NIL, else false. EQUAL(X, Y) which is true if X
and Y are the same atoms or equivalent lists, else false. Which operations below
extracts a from the list A = (be(a(cat))), assuming H HEAD and T TAIL?

甲. `H(T(T(H(T(A))))))

乙. `H(T(T(A)))

丙. `H(T(T(H(T(A)))))

丁. `H(T(H(T(H(T(A))))))

10. (3pts) As in question 9, which function below implements the APPEND operation
that concatenates two lists into one?

甲. `List APPEND (List a, List b) {
   return CONS(HEAD(a), APPEND(TAIL(a), b));
}

乙. `List APPEND (List a, List b) {
   return NULL(b) ? a : APPEND(CONS(a, HEAD(b)), TAIL(b));
}

丙. `List APPEND (List a, List b) {
   return NULL(a) ? b : APPEND(CONS(HEAD(a), APPEND(TAIL(a), b));
}

丁. `List APPEND (List a, List b) {
   return NULL(a) ? b : APPEND(TAIL(a), CONS(HEAD(a), b));
}
11. Given a preorder traversal ABCDEFGHIJ:
   (a) (2 pts) Does such a traversal sequence uniquely define a binary tree?
   (b) (3 pts) Now suppose you are also given another inorder traversal
           CBEDAHGFJIJ. Does such a pair of traversal sequences, i.e.
           preorder and inorder, uniquely define a binary tree?
   (c) (5 pts) Construct a possible binary tree that corresponds to the preorder and
           the inorder sequences.

12. Draw the internal memory representation of the following binary tree, using
    甲、(2 pts) array representation (i.e. sequential representation)
    乙、(2 pts) linked representation
    丙、(2 pts) Write the inorder and the level-order traversal sequences.

13. (6 pts) For any nonempty binary tree, prove that if \( n_0 \) is the number of leaf nodes
    and \( n_2 \) the number of nodes of degree 2, then \( n_0 = n_2 + 1 \).

14. Given the following undirected graph:

   甲、(3 pts) show its adjacency matrix
   乙、(3 pts) show its adjacency list
丙、(3 pts) show its sequential representation, using a one-dimensional array
丁、(3 pts) show the depth-first spanning tree, starting from node 0 and in the
alphabetical order.
戊、(3 pts) show the breadth-first spanning tree, starting from node 0 and in the
alphabetical order.

FIGURES ARE IN THE NEXT PAGE.

15. (5pts) Given is a min-max heap as shown in Figure 1. Consider the insertion
operation in a min-max heap. Suppose we are insertion a key $j$ into that min-max
heap. We know that $j$ is first inserted as a right son of 10. To move $j$ to its
correct location, we first compare $j$ against 10. Show that if $j > 10$, the final
location of $j$ will be in the max-level.

16. (a.) (5pts) Given is an $n$ internal nodes binary search tree (so there are $n-1$ external
nodes). Suppose all the nodes in this binary search tree have equally
probability to be accessed. What is the cost for the optimal binary search tree?
(What is your reason?)
(b.) (5pts) If all the nodes (internal nodes and external nodes) have different
access probability, try your best to explain how to establish the optimal binary
search.

17. (3pts) What is the resulted AVL-tree after insertion 4 into the AVL tree shown in
Figure 2.

18. (5pts) What are the 2-3-trees after (a.) first insertion 15, (b) then insertion 12 into
the 2-3-tree shown in Figure 3.

19. (4pts) When we say $\Theta(n \log n) + O(n \log n)$, what is result after such addition?
What are your reasons?

20. Answer TRUE or FALSE. (6pts)
甲、Quick sort has the best performance in sorting $n$ numbers. It takes average
$O(n \log n)$ time. Thus the best possible sorting algorithm needs $\Omega(n \log n)$
comparisons to sort $n$ numbers.
乙、$\Theta(n \log n)$ can better describe the time required to merge-sort $n$ numbers
than $O(n \log n)$.
丙、$\Theta(n^2)$ can better describe the time required to selection-sort $n$ numbers
than $O(n^2)$.