編號： 181

國立成功大學 106 學年度碩士班招生考試試題

系　　所：電機工程學系

考試科目：資料結構

考試日期：0213，節次：2

※ 考生請注意：本試題不可使用計算機。　請於答案卷(卡)作答，於本試題紙上作答者，不予計分。

一、 複選題: (20 分, 全對才給分, 每題 5 分)

1. Which of the following statement(s) is (are) true?

(A) If $g_1(n) \in \Omega(f_1(n))$ and $g_2(n) \in \Omega(f_2(n))$ then $g_1(n)+g_2(n) \in \Omega(\max\{f_1(n)+f_2(n)\})$

(B) $5\log n^2 + \log\log n^2$ is $\Theta(\log n^2)$

(C) $f(n) \in \Theta(g(n))$ if and only if $f(n) \in O(g(n))$ and $f(n) \in \Omega(g(n))$

(D) $f(n) \in O(g(n))$ and $h(n) \in \Omega(g(n))$ imply $f(n) \in O(h(n))$

2. Which of the following statement(s) is (are) true?

(A) A binary tree is a special case of a tree.

(B) It is still legal to have a tree without any node.

(C) Given a root n of a tree T_1 whose left subtree and right subtree are respectively S_1 and S_2, we generate a new tree T_2 by exchanging S_1 and S_2. Then, T_1 is equal to T_2.

(D)The maximum number of nodes in a binary tree is 2^{K-1} if the depth of a tree is k.

3. Which of the following statement(s) is (are) true?

(A) A depth-first algorithm is implemented by queue.

(B) The time complexity of the Prime algorithm is O(E logV) if it is implemented by an array.

(C) There exists no back edge in a spanning tree generated by a breath-first algorithm.

(D) A max heap can be implemented by a complete binary tree.

4. Which of the following statement(s) is (are) true?

(A) The insertion sort is a stable algorithm.

(B) The quick sort is the fastest sorting algorithm.

(C) The merge sort and quick sort both apply the divide-and-conquer strategy, but the quick sort is the in-place algorithm while the merge sort is not.

(D) The heap sort is a comparison based sorting algorithm.

國立成功大學 106 學年度碩士班招生考試試題

系　　所：電機工程學系

考試科目：資料結構

考試日期：0213，節次：2

二、簡答題: (60 分)

1. (20 **pts**) Give a sequence of numbers 6, 2, 4, 7, 3, 9, 5 recorded by an array (the first index of the array is zero).

   a. (6 **pts**) Sort the numbers in the increasing order by the quick sort algorithm and show the status of the array each time after you change the location of pivot (**take the last digit as pivot**). Each time the array [0...n-1] is divided into two subarrays whose indices are [0 ... $\left\lfloor\frac{n-1}{2}\right\rfloor$] and [$\left\lfloor\frac{n-1}{2}\right\rfloor + 1$ ... n-1], respectively, and the subarray with small index is handled first.

   b. (4 **pts**) Show the condition when the algorithm has to take the longest time according to the above numbers. What is the time complexity if there exist n numbers and why?

   c. (5 **pts**) Show the status of the heap in an array before you want to sort the number **in the decreasing order.**

   d. (5 **pts**) Based on the result of b, you delete the root and exchange the root with the number in the array with the largest index. Please show the status of the array after you legal the new heap.

2. (20 **pts**) **Biconnected Components**

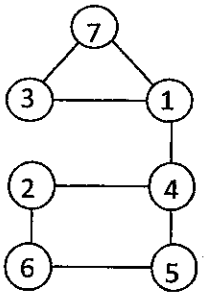
   a. (3 **pts**) Show the depth-first search tree.

   b. (7 **pts**) List the dfs numbers and low values of all nodes in a table. (assume you start with node 1 and the search always start from a node with small number).

   c. (4 **pts**) Show all articulation points and explain why they are the articulation points.

   d. (3 **pts**) Give the definition of the back edge and show backward edges if the figure has.

   e. (3 **pts**) Give the definition of the forward edge and show forward edges if the figure has.



3. (20 **pts**) **Threaded binary tree.**

   a. (2 **pts**) How many threads are there in a threaded binary tree which contains n nodes?

   b. (5 **pts**) Obtain the binary tree whose preorder traversal is ABCEDF and inorder traversal is CBDEAF.

   c. (3 **pts**) Show the components in the data structure of a node in a thread binary tree and explain their purposes.

   d. (5 **pts**) Show the post-threaded binary tree corresponding to the binary tree generated by b. Make sure that the tree does not contain any dangling thread.

   e. (5 **pts**) Show the data after calling the function **OPT** if current node points to the node B in the tree.

```
T* ThreadInorderIterator:: OPT() {
   ThreadNode <T> *temp = currentNode->rightChild;
   if (!currentNode->rightThread)
        while (!temp->leftThread) temp=temp->leftChild;
   currentNode = temp;
   if (currentNode == root) return 0;
   else return &currentNode->data;
}
```

三、填充題 (20 分 ，每小題 2 分)

The following shows a C++ implementation of a forward iterator for the class Chain, and ChainIterator is a public member of Chain.

```
class ChainIterator {
public:
    // increment
    ChainIterator& operator ++() { // preincrement
        current = current->link; // increment iterator
        return *this; } // reference return
    ChainIterator operator ++(int)  { // postincrement
        ChainIterator old = *this; // hold current state of object
        current = current->link; // increment iterator
        return old; } // value return; return unincremented/saved object
private:
    ChainNode *current;
};
```

(-) (2 pts) What is the purpose of the iterator used by a container in C++?

(=) (10 pts) Assume that we use the definition of **class** ChainNode and the chain nodes shown in the following figure, where first is a ChainNode pointer pointing to the first node of the chain. Please implement the following predecrement and postdecrement function for ChainIterator.

```
class ChainNode {
private:
    int data;
    ChainNode *link;
};
```


```
ChainIterator& operator –() { //predecrement
  if (current == first) {___A___;}
  ChainNode *it = first;
  while (___B___) {
      _____C_____;
  }
  _____D_____;
  return  *this;
}
```

```
ChainIterator operator --(int) { //postdecrement
    if (current == first){___A___;}
    ChainNode *it = first;
    ChainIterator old ;
    while (___B___) {
        _____C_____;
    }
    _____E_____;
    _____D_____;
    return  old;
}
```

國立成功大學 106 學年度碩士班招生考試試題

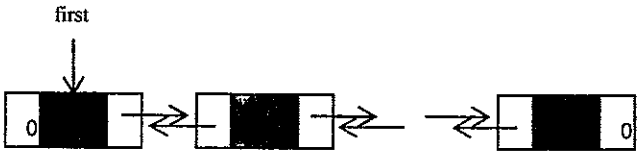
系　　所：電機工程學系

考試科目：資料結構

考試日期：0213，節次：2

〈三〉 (4 **pts**) If we change the definition of ChainNode and the way of nodes connecting in the chain as follows, please implement the corresponding predecrement and postdecrement function for ChainIterator.

```
class ChainNode {
private:
    int data;
    ChainNode *left, *right;
};
```




```
ChainIterator& operator –() { //predecrement
    _____F_____;
    return   *this;;
}
```

```
ChainIterator operator --(int) { //postdecrement
    ChainIterator old;
    _____G_____;
    _____F_____;
    return   old;
}
```

〈四〉 (4 **pts**) The iterator supports the dereference operator and the inequality operator. Please complete the codes in the following.

```
// dereferencing operators
T& operator *() const {return_____H_____;}
bool operator!=(const ChainIterator right) const
{return_____I_____;}
```