华东师范大学期末试卷（A）

2018—2019 学年第二学期

课程名称：数据结构与算法

学生姓名：_____ 学　　号：_____

专　　业：_____ 年级/班级：_____

课程性质：专业必修

一	二	三	四	五	六	七	八	总分	阅卷人签名

一，填空题，补全如下程序代码（12 分，每空 2 分）

1. 根据课本内容补全如下 Buildable_tree 的 connect_trees 程序代码。

```
template <class Record>
void Buildable_tree<Record> :: connect_trees(
const List < Binary_node<Record>* > &last_node)
/* Pre: The nearly-completed binary search tree has been initialized. List last_node
has been initialized and contains links to the last node on each level of the tree.
Post: The final links have been added to complete the binary search tree. */
{
    Binary_node<Record>
        *high_node, // from last node with NULL right child
        *low_node; // candidate for right child of high node
    int high_level = last_node.size( ) - 1,    low_level;
    while (____【1】____) {
        last_node.retrieve(high_level, high_node);
        if (high_node->right != NULL)
            high_level--; // Search down for highest dangling node.
        else { // Case: undefined right tree
            low_level = high_level;
            do { // Find the highest entry not in the left subtree.
                last_node.retrieve(--low_level, low_node);
            } while (low_node != NULL &&___【2】___);
            high_node->right = low_node;
            high_level = low_level;
        }
    }
}
```

2. 根据课本内容补全如下汉诺塔的搬移程序代码。

```
void move(int count, int start, int finish, int temp)
{
    if (count > 0) {
        ____【3】____
        cout << "Move disk " << count << " from " << start
             << " to " << finish << "." << endl;
        ____【4】____
    }
}
```

3. 根据课本内容补全如下 Trie 的插入 insert 程序代码。

```
Error_code Trie::insert(const Record &new_entry)
/* Post: If the Key of new_entry is already in the Trie , a code of duplicate_error is
returned. Otherwise, a code of success is returned and the Record new entry is
inserted into the Trie . */
{
    Error_code result = success;
    if (root == NULL) root = new Trie_node; // Create a new emptyTrie .
    int position = 0; // indexes letters of new entry
    char next_char;
    Trie_node *location = root; // moves through the Trie
    while (next_char = new_entry.key_letter(position) != '\0') {
        int next_position = alphabetic_order(next_char);
        if (___【5】___)location->branch[next_position] = new Trie_node;
        location = location->branch[next_position];
        position++;
    }
    if (___【6】___) result = duplicate_error;
    else location->data = new Record(new_entry);
    return result;
}
```

二，程序分析题（20 分，每题 5 分）

1. 根据如下递归程序代码，给出表达式 Func(12, 15)的值。

```
int Func( int i,  int j ) {
    if (i < 11)
        if (j < 11)
            return i + j;
        else
            return j + Func(i, j - 2);
    else
        return i + Func(i - 1, j);
}
```

2. 根据如下程序代码，给出函数 Quiz(2)的输出结果。

```
void Quiz( int n )
{
    if (n > 0)
    {
        cout << 0;
        Quiz(n - 1);
        cout << 1;
        Quiz(n - 1);
    }
}
```

3. 根据如下程序代码，给出函数 Func (75)的输出结果。

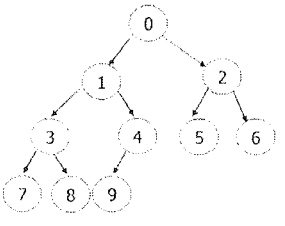
```
void Func( /* in */ int n )
{
    if (n > 0)
    {
        Func(n / 8);
        cout << n % 8;
    }
}
```

4. 给出如下程序片段的输出结果。

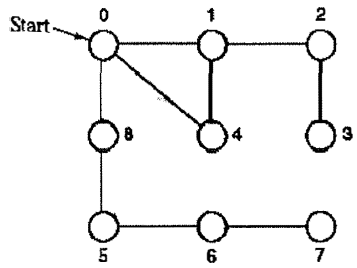
```
int item;
MyStack<int> numbers;
numbers.push(1);
numbers.push(3);
numbers.push(2);
numbers.pop();
numbers.pop();
numbers.top(item);
cout<<item;
MyQueue<int> numbers;
for (int i = 1; i < 5; i++) numbers.append(i);
numbers.serve();
numbers.serve();
numbers.retrieve(i);
cout<<i;
```

三，简单题（48 分，每题 8 分）

1. 请解释二叉树的前序遍历，中序遍历和后序遍历。根据下图分别给出其前序遍历，中序遍历和后序遍历的序列。



2. 分别给出下图从顶点 0 出发的深度优先遍历序列和广度优先遍历序列。

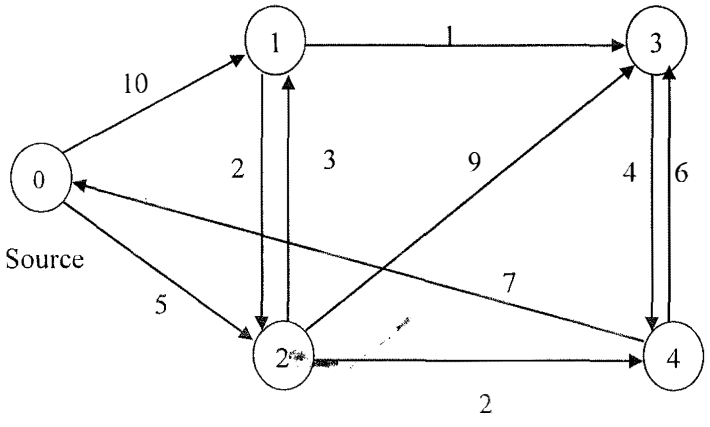


3. 将如下的序列插入到一棵空的 AVL 树中后，给出该树的前序，中序，后序。
A，Z，B，Y，C，X，D，W，E，V，F

4. 解释 B 树的定义，并给出如下序列插入一颗空的序为 5 的 B 树的图示过程。
a, g, f, b, k, d, h, m, j, e, s, i, r, x, c, l, n, t, u

5. 给出如下序列插入一个空的小根堆（根最小）的图示过程。插入完成后，并给出删除根以后的新堆的形态。
10, 12, 1, 14, 6, 5, 8, 15, 3, 9, 4, 11, 13, 2

6. 根据如下的有向图，给出从顶点 0 分别到其他顶点的最短路径以及数值。



四，编程题（20 分，每题 10 分）

1. 设有一个表头指针为 h 的单链表。试设计一个算法，通过遍历一趟链表，将链表中所有结点的链接方向逆转。逆转后，链表的表头指针 h 指向原链表的最后一个结点。

```
void Inverse (struct Node ** h);
```

2. 指针 f 指向一个非空单链表的表头，单链表的结点结构 Node 如下，采用递归的方法分别实现如下函数。

```
struct Node {
    int data;
    Node *next;
};

int Max (Node *f ) ;  //return the max value in the link list.
int Num (Node *f );   //return the number of the nodes in the link list
```