中 山 大 学

二〇一一年攻读硕士学位研究生入学考试试题

科目代码： 874

科目名称： 数据结构（A）

考试时间： 1 月 16 日 下 午

考 生 须 知

全部答案一律写在答题纸上，答在试题纸上的不计分！请用蓝、黑色墨水笔或圆珠笔作答。答题要写清题号，不必抄题。

一、算法的时间复杂度问题（20 分）

(1) 假设数组 A 含有 n 个元素，函数 Random(n)需花费常数时间，sort(A,n)需花费 $nlog_2n$ 时间，求下面程序段的时间复杂度 T(n)。

```
sum=0;
for(int i=0; i<n*n; i++)
        sum++;
for (i=0; i<n; i++){
    for(j=0; j<n; j++)
        A[i]=Random(n);
    sort(A, n);
}
```

(2) 假设 int 类型的数组 a 含有 n+1 个元素，函数 f 如下定义，那么，求 f(0)的时间复杂度 T(n)。

```
void f(int j ){
    int temp;
    if ( j<n) {
        if(j!=0)for(int i=j;i<=n;i*=2){temp=a[i];a[i]=a[j];a[j]=temp;}
        j++;
        f(j);
    }
}
```

二、（34 分）关于对象的定义问题(注释你写的代码)

(1) 请给出树的 3 种存储表示方法的 C 语言定义代码；

(2) 请给出图的 2 存储表示方法的 C 语言定义代码；

三、（20 分）已知哈希表地址空间为 0~8，哈希函数为 H(key)=key % 7。 若向表中依次插入关键字（100，20，21，35，3，78，99，45），请列出冲突解决方法采用线性探测再散列时插入完成后的哈希表中的情况（图示），并计算查找成功和不成功时的平均查找长度。

四、（20 分）已知一字符串 bcbdebcecbdcabd，试设计其赫夫曼编码并画出相应的赫夫曼树。

五、（30 分）已知有向图的顶点集合为{A, B, C, D, E, F}，有向边的集合为{<A, B>, <B, F>, <F, D>, <C, A>, <C, D>, <C, E>, <C, F>, <E, D>, <E, F> }。请回答下面问题：

(1) 画出该有向图；

(2) 写出该图从顶点 C 开始的深度遍历和广度遍历序列；

(3) 写出它的拓扑排序序列。

六、（26 分）代码填空

(1) 下面是仅给出了部分操作的下面给出了带头结点的线性链表的定义和实现。试在程序的每一划线部分填入一条语句或表达式，完成相应的操作。

```
typedef struct node{
   int elem;
   struct node *next;
}node,*LinkListPtr;
typedef struct {
   LinkListPtr head,tail; //表头，表尾指针
   LinkListPtr curr;      //直接或间接指向当前结点的指针
} LinkList;

void InitLinkList(LinkList &L) //初始化线性表
{
     L.head = (node *)malloc(sizeof(node));
     if(L.head == NULL){cout<<"Insufficient memory available\n" ;exit(0);};
     L.tail = L.head; L.curr = L.head;
}

void insert(LinkList &L,int & item) //在表 L 的当前位置处插入元素 item
{
   if(L.curr == NULL){cout<<"Current position is not a legal position\n";exit(0);   };
   node * newnode = (node *)malloc(sizeof(node));
   if(newnode == NULL){cout<<"Insufficient memory available\n" ;exit(0);};
    newnode->elem=item;
    ____①____;
    ____②____;
   if (L.tail == L.curr) L.tail = L.curr->next;
}
void setFirst(LinkList &L)     //将表的当前位置定位于第 1 个元素
{
    ____③____;
}

int currValue(LinkList &L)   //将表的当前位置的元素值返回
{
    if(!isInList(L))
    {
        cout<<"Current position is not a legal position\n";exit(0);
    }
     return ____④____;
}
```

```
bool isEmpty(LinkList &L) //判断线性表是否为空
{
      ⑤                          ;
}

bool isInList(LinkList &L) //判断 curr 是否在表中
{
    return (L.curr != NULL) && (L.curr->next != NULL) && (!isEmpty(L));
}

bool find(LinkList &L ,const int & eval) //从表的当前位置开始查找元素 eval
{
  while (isInList(L))
    if (    ⑥                  ) return TRUE;
    else    ⑦                          ;
  return FALSE;
}
```

(2) 下面是仅给出了部分操作的二叉树类的定义和实现。试在程序的每一划线部分填入一条语句或表达式，完成计算度为 1 的结点个数的操作 countNodes1( )。

```
typedef struct BinNode{
      int data;
      struct BinNode* leftchild, * rightchild;
}BinNode, *BinTree;

void InitBinTree(BinTree &root)   //初始化二叉树 root
{
    root=Null;
}
void DestroyBinTree(BinTree &root)   //销毁二叉树 root
{
    if (root == NULL) return;
    DestroyBinTree(root->leftchild);
    DestroyBinTree (root->rightchild);
    free(root);
}
int countNodes1(BinTree &root)   //计算二叉树 root 中度为 1 的结点数
{
    if (    ①                                                  ){
        if(root-> leftchild!=NULL && root-> rightchild!=NULL)
            ②                ;
        else   if(root-> leftchild==NULL &&  ③                )
          ④                          ;
     else if (root-> leftchild!=NULL &&   ⑤                )
            ⑥                    ;
   }
    return 0;
}
```