《数据结构（A类）》（A卷）参考答案

1. 单项选择题（每格1.5分，共24分）

1.D 2.C 3.C 4.D 5.A 6.D 7.B 8.C 9.B 10.D 11.B 12.C 13.C 14.B 15.C 16.B

1. 程序填充题（每格1.5分，共24分）
   1. 1）p->data > x

2）f->left = p->right

3）delete p

4）s->right

5）s == p->left

6）delete s

* 1. 1）(low + high) / 2

2）high = mid - 1

3）low = mid + 1

* 1. 1）-1（与第二个空格对应，如-1对应99、0对应100）

2）stack.top == 99

3）++stack.top

4）stack.s[stack.top] = x

* 1. 1）p == NULL

2）p或p != NULL

3）q

1. 简答题（每题8分，共24分）
   1. 每个序列各2分，每个图各1分，共8分。

DFS序列：ABEDFCGHJI

BFS序列：ABCEFDGHJI

* 1. 哈夫曼树得4分，10个字符的哈夫曼编码得4分（每个0.4分）。

**构造哈夫曼树的要求：对于新构造的结点，其左儿子的权值较小，右儿子的权值较大。**

! : 010010 @ : 0100111 # : 01011 $ : 1

% : 0100110 ^ : 0110 & : 01010 \* : 00

( : 01000 ) : 0111

* 1. 两小题各4分。

1）35：探测序列9、1、2，共3次

20：探测序列7、4，共2次

33：探测序列7，共1次

48：探测序列9，共1次

2）平均比较长度：(3+2+1+1)/4=7/4 或 (3+2+1+1+2)/5=9/5

1. 分析题（每题8分，共16分）
   1. 1）2分：排序

2）3分：++x，--y

3）3分：最坏情况*O(n2)*，最好情况*O(nlog2n)*

* 1. 1）2分：p指向最后一个结点

2）3分：将第一个结点作为链表最后一个结点

3）3分：(a2, a3, …, an, a1)

1. 设计题（12分）

1）设计思路（8分）

a）2分：首先定义一个大小为k的最小化堆，用数组的前k个元素组成一个最小化堆

b）2分：对剩余的数组中的每个元素进行如下处理：如果当前元素比堆顶元素大，则删除堆顶元素，添加当前元素进堆

c）2分：最后堆顶元素即为第k大值。创建一个大小为k堆时间复杂度为O(k)

d）2分：在一个大小为k的堆中添加或删除元素时间复杂度都是O(logk)，所有总的时间复杂度为O(k)+(N-k)O(logk)=O(Nlogk)

2）伪代码（4分）

**template** <**typename** T>

T Getkth(T a[], **int** N, **int** K) {

//利用优先队列取数组a前k个元素建立一个最小化堆H;

**for** (**int** i = k; i < N; i++) { //for循环（3分）

**if** (H的堆顶元素<a[i]) {

删除堆顶元素；a[i]进堆；

}

}

**return** 堆顶元素； //返回值（1分）

}

1. 附加题（10分）

//判断顶点vi和vj之间是否有路径，有则返回true，否则返回false。

**bool** path\_i\_j(TypeOfVer vi, TypeOfVer vj) {

**int** i, j, k, m;

**for** (i=1; i<=Vers; i++) visited[i]=0; //访问标记数组初始化

初始化4分

**for** (i=0; i<Vers; ++i) **if** (verList[i].ver==vi) break; //查找vi的编号

**if** (i==Vers) {**cout**<<"顶点vi不存在\n"; **return** false;}

**for** (j=0; j<Vers; ++j) **if** (verList[j].ver==vj) break; //查找vj的编号

**if** (j==Vers) {**cout**<<"顶点vj不存在\n"; **return** false;}

**int** \*s=**new** **int**[Vers], top=0;

s[++top]=i; //编号i入栈

edgeNode \*p;

**while** (top>0) {k=s[top--]; //出栈（2分）

p=verList[k].head;

**while** (p!=NULL && visited[p->end]==1) //（2分）

p=p->next; //查第k个链表中第一个未访问的边结点

主体6分

**if** (p==NULL) top--;

**else** {m=p->end;

**if** (m==j) **return** true; //vi和vj 间有路径（1分）

**else** {visited[m]=1; s[++top]=m;}

}

}

**return** false; //顶点vi和vj 间无通路（1·分）

}