一、填空题（5分每题，一共8题）  
1、两个人A（速度为a）、B（速度为b）在一直路上相向而行。在A、B距离为s的时候，A放出一个鸽子C（速度为c），C飞到B后，立即掉头飞向A，遇到A在掉头飞向B......就这样在AB之间飞来飞去，直到A、B相遇，这期间鸽子共飞行路程为？  
答案是：s\*c/(a+b)  
  
2、（he）的平方=she。h、e、s代表的数字？  
答案是：分别代表2、5、6  
  
3、运算（93&-8）的结果为：88  
  
4、将一个无序整数数组构造成一个最大堆，最差时间复杂度为：  
5、int \*p = &n;  
那么\*p的值是（）  
A、p的值     B、p的地址     C、n的值     D、n的地址  
  
6、一个完全二叉树有770个节点，那么其叶子的个数为：385  
  
7、有一个二维数组a[1...100 , 1...65]有100行，65列，我们以行序为主序，如果该数组的基地址是10000，且每个元素占2个存储单元，请问a[56 , 22]的存储地址是：17194  
8、以下代码输出结果是：

**[cpp]** [view plaincopyprint?](http://blog.csdn.net/hackbuteer1/article/details/8484974)

1. class B
2. {
3. public:
4. B()
5. {
6. cout<<"B constructor\n";
7. s = "B";
8. }
9. void f()
10. {
11. cout<<s;
12. }
13. private:
14. string s;
15. };
17. class D : public B
18. {
19. public:
20. D() : B()
21. {
22. cout<<"D constructor\n";
23. s = "D";
24. }
25. void f()
26. {
27. cout<<s;
28. }
29. private:
30. string s;
31. };
33. **int** main(void)
34. {
35. B \*b = new D();
36. b->f();
37. ((D\*)b)->f();
38. delete b;
39. return 0;
40. }

class B

{

public:

B()

{

cout<<"B constructor\n";

s = "B";

}

void f()

{

cout<<s;

}

private:

string s;

};

class D : public B

{

public:

D() : B()

{

cout<<"D constructor\n";

s = "D";

}

void f()

{

cout<<s;

}

private:

string s;

};

int main(void)

{

B \*b = new D();

b->f();

((D\*)b)->f();

delete b;

return 0;

}

输出结果是  
B constructor  
D constructor  
BD  
  
二、编程题  
1、数组乘积（15分）  
输入：一个长度为n的整数数组input  
输出：一个长度为n的整数数组result，满足result[i] = input数组中除了input[i]之外所有数的乘积（假设不会溢出）。比如输入：input = {2,3,4,5}，输出result = {60,40,30,24}  
程序时间和空间复杂度越小越好。  
C/C++：  
int \*cal(int\* input , int n);  
  
Java:  
int[] cal(int[] input);

**[cpp]** [view plaincopyprint?](http://blog.csdn.net/hackbuteer1/article/details/8484974)

1. **int** \*cal(**int**\* input , **int** n)
2. {
3. **int** i ;
4. **int** \*result = new **int**[n];
5. result[0] = 1;
6. for(i = 1 ; i < n ; ++i)
7. result[i] = result[i-1]\*input[i-1];
8. result[0] = input[n-1];
9. for(i = n-2 ; i > 0 ; --i)
10. {
11. result[i] \*= result[0];
12. result[0] \*= input[i];
13. }
14. return result;
15. }

int \*cal(int\* input , int n)

{

int i ;

int \*result = new int[n];

result[0] = 1;

for(i = 1 ; i < n ; ++i)

result[i] = result[i-1]\*input[i-1];

result[0] = input[n-1];

for(i = n-2 ; i > 0 ; --i)

{

result[i] \*= result[0];

result[0] \*= input[i];

}

return result;

}

2、异形数（25分）  
在一个长度为n的整形数组a里，除了三个数字只出现一次外，其他的数字都出现了2次。请写程序输出任意一个只出现一次的数字，程序时间和空间复杂度越小越好。  
例如： a = {1,3,7,9,5,9,4,3,6,1,7}，输出4或5或6  
C/C++：  
void find(int\* a , int n);  
  
Java:  
void find(int[] a);

**[cpp]** [view plaincopyprint?](http://blog.csdn.net/hackbuteer1/article/details/8484974)

1. // lowbit表示的是某个数从右往左扫描第一次出现1的位置
2. **int** lowbit(**int** x)
3. {
4. return x&~(x-1);
5. }
7. void find(**int**\* a , **int** n)
8. {
9. **int** i , xors;
10. xors = 0;
11. for(i = 0 ; i < n ; ++i)
12. xors ^= a[i];
13. // 三个数两两的异或后lowbit有两个相同，一个不同，可以分为两组
14. **int** fips = 0;
15. for(i = 0 ; i < n ; ++i)
16. fips ^= lowbit(xors ^ a[i]);
17. // 表示的是：flips=lowbit(a^b)^lowbit(a^c)^lowbit(b^c)
18. **int** b;    // 假设三个只出现一次的其中一个数为b
19. b = 0;
20. for(i = 0 ; i < n ; ++i)
21. {
22. if(lowbit(xors ^ a[i]) == fips)
23. b ^= a[i];
24. }
25. // 成功找到三个数中一个数
26. cout<<b<<endl;
27. }

// lowbit表示的是某个数从右往左扫描第一次出现1的位置

int lowbit(int x)

{

return x&~(x-1);

}

void find(int\* a , int n)

{

int i , xors;

xors = 0;

for(i = 0 ; i < n ; ++i)

xors ^= a[i];

// 三个数两两的异或后lowbit有两个相同，一个不同，可以分为两组

int fips = 0;

for(i = 0 ; i < n ; ++i)

fips ^= lowbit(xors ^ a[i]);

// 表示的是：flips=lowbit(a^b)^lowbit(a^c)^lowbit(b^c)

int b; // 假设三个只出现一次的其中一个数为b

b = 0;

for(i = 0 ; i < n ; ++i)

{

if(lowbit(xors ^ a[i]) == fips)

b ^= a[i];

}

// 成功找到三个数中一个数

cout<<b<<endl;

}

3、朋友圈（25分）  
假如已知有n个人和m对好友关系（存于数字r）。如果两个人是直接或间接的好友（好友的好友的好友...），则认为他们属于同一个朋友圈，请写程序求出这n个人里一共有多少个朋友圈。  
假如：n = 5 ， m = 3 ， r = {{1 , 2} , {2 , 3} , {4 , 5}}，表示有5个人，1和2是好友，2和3是好友，4和5是好友，则1、2、3属于一个朋友圈，4、5属于另一个朋友圈，结果为2个朋友圈。  
最后请分析所写代码的时间、空间复杂度。评分会参考代码的正确性和效率。  
C/C++：  
int friends(int n , int m , int\* r[]);  
  
Java:  
int friends(int n , int m , int[][] r);

**[cpp]** [view plaincopyprint?](http://blog.csdn.net/hackbuteer1/article/details/8484974)

1. // 简单的并查集应用
2. **int** set[10001];
4. inline **int** find(**int** x)           //带路径优化的并查集查找算法
5. {
6. **int** i , j , r;
7. r = x;
8. while(set[r] != r)
9. r = set[r];
10. i = x;
11. while(i != r)
12. {
13. j = set[i];
14. set[i] = r;
15. i = j;
16. }
17. return r;
18. }
19. inline void merge(**int** x , **int** y)     //优化的并查集归并算法
20. {
21. **int** t = find(x);
22. **int** h = find(y);
23. if(t < h)
24. set[h] = t;
25. else
26. set[t] = h;
27. }
29. **int** friends(**int** n , **int** m , **int**\* r[])
30. {
31. **int** i , count;
32. for(i = 1 ; i <= n ; ++i)    //初始化并查集，各点为孤立点，分支数为n
33. set[i] = i;
34. for(i = 0 ; i < m ; ++i)
35. merge(r[i][0] , r[i][1]);
36. count = 0;
37. for(i = 1 ; i <= n ; ++i)
38. {
39. if(set[i] == i)
40. ++count;
41. }
42. return count;
43. }

// 简单的并查集应用

int set[10001];

inline int find(int x) //带路径优化的并查集查找算法

{

int i , j , r;

r = x;

while(set[r] != r)

r = set[r];

i = x;

while(i != r)

{

j = set[i];

set[i] = r;

i = j;

}

return r;

}

inline void merge(int x , int y) //优化的并查集归并算法

{

int t = find(x);

int h = find(y);

if(t < h)

set[h] = t;

else

set[t] = h;

}

int friends(int n , int m , int\* r[])

{

int i , count;

for(i = 1 ; i <= n ; ++i) //初始化并查集，各点为孤立点，分支数为n

set[i] = i;

for(i = 0 ; i < m ; ++i)

merge(r[i][0] , r[i][1]);

count = 0;

for(i = 1 ; i <= n ; ++i)

{

if(set[i] == i)

++count;

}

return count;

}