一、单项选择题（每题2分，共计30分）

1、以下说明不正确的是（）。B

A．冯诺伊曼被誉为“电子计算机之父”。

B．图灵的“图灵机”是第一台可编程的电子计算机。

C．香农是现代信息论的著名创始人，信息论及数字通信时代的奠基人。

D．西蒙是人工智能中影响最大的符号主义学派的创始人和代表人物。

2、以下四项中（ ）项与别的选项最不相同。A

A．P∧﹁ Q

B．（P ∧ Q）→ P

C．P →（ P ∨ Q）

D．﹁（ P ∨ Q）∨﹁（ ﹁P ∧ ﹁Q）

3、在计算机的四个发展阶段中，以下哪个是第二个阶段：（）。A

A．晶体管时代

B．电子管时代

C．小规模集成电路时代

D．大规模和超大规模集成电路时代

4、真正能唯一标识出一台计算机网络中一台计算机的地址是（）。D

A．地址掩码

B．ip地址

C．域名地址

D．MAC地址

5、最大容量为n的循环队列，队尾指针是rear，队头是front，则队空的条件是（　）。B

A. (rear+1)%n==front

B. rear==front

C．rear+1==front

D. (rear-l)%n==front

6、30！的结果，转换成3进制数字结尾有（）个0？D

A．7

B．10

C．12

D．14

7、567\*456=150216在n进制下成立，问n为（）。B

A．9

B．18

C．21

D．39

8、对于下面的一组函数：

void fun (int a, float x); // 函数1

void fun (int a, int x); // 函数2

void fun (float a, float x ); // 函数3

void fun (float a, int x ); // 函数4

void fun (int a[], int n); // 函数5

void fun (int \*p, int n); // 函数6

以下说法不正确的是（）。 D

A．函数1和函数2可以构成重载函数；

B．函数1和函数4可以构成重载函数；

C．函数3和函数4可以构成重载函数；

D．函数5和函数6可以构成重载函数；

9、以下关于渐进记号的性质是正确的有：（） A

A．f(n) =Θ(g(n)),g(n) =Θ(h(n)) ⇒f(n) =Θ(h(n))

B．f(n) =O(g(n)),g(n) =O(h(n)) ⇒h(n) =O(f(n))

C．O(f(n))+O(g(n)) = O(min{f(n),g(n)})

D．f(n) = O(g(n)) ⇔g(n) = O(f(n))

10、由权值分别为3,8,6,2,5的叶子结点生成一棵哈夫曼树，它的带权路径长度为（ ）。D

A. 24 B. 48 C. 72 D. 53

11、T（n）表示当输入规模为n时的算法效率，以下算法效率最优的是() C

A．T（n）= T（n–1）+1，T（1）=1

B．T（n）= 2n^2

C．T（n）= T（n/2）+1，T（1）=1

D．T（n）= 3nlogn

12、已知一个有向图的边集为{<a,b>,<a,c>,<a,d>,<b,d>,<b,e>,<d,e>}，则由该图产生的一种可能的拓扑序列为( )。 A

A. a,b,c,d,e B. a,c,d,e,b C. a,c,b,e,d D. a,c,d,b,e

13、Kathy函数是这样定义的：

f(1)=1

f(3)=3

f(2n)=f(n)

f(4n+1)=2f(2n+1)-f(n)

f(4n+3)=3f(2n+1)-2f(n)

对于一个给定的数m=55，求出所有满足f(n)=n,(n<=m)的自然数n的个数（ ） B

A. 11 B. 13 C. 17 D. 25

14、一个游戏：连续抛掷硬币，直到最近三次硬币抛掷结果是“正反反”或者“反反正”。如果是前者，Alice赢，如果是后者Bob赢，问Alice对Bob的胜率：（ ）。 D

A. 1:1 B. 1:3 C. 3:2 D. 3:1

15、下列说法正确的是（ ）。C

A．2020年开始，NOIP将不再支持Pascal语言和C语言。

B．NOI中选手可以使用自己的头文件，头文件必须包含在选手文件根目录下。

C．选手在NOIP考试过程中，不得改变文件系统的访问权限。

D．在NOI考试结束提示发出后，考生应该关闭计算机有序离场，不能吵到正在补时的考生。

三、阅读程序写结果（共3题，每题10分，共计30分）

1、

1 #include<bits/stdc++.h>

2 using namespace std;

3 int a[100005],b[100005],c[100005];

4 int main() {

5 int n,i,j,m,k=0,sign=0;

6 cin>>n;

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

8 cin>>a[i];

9 c[i]=a[i];

10 }

11 sort(c,c+n);

12 for(j=0; j<n-1; ++j) {

13 if(a[j]==c[n-1]) {

14 k=n-1-j;

15 }

16 if(a[j+1]<a[j])

17 sign=1;

18 }

19 if(sign==0) {

20 cout<<0<<endl;

21 return 0;

22 }

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

24 b[(i+k)%n]=a[i];

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

26 if(b[i]!=c[i]) {

27 cout<<-1<<endl;

28 return 0;

29 }

30 }

31 cout<<k<<endl;

32 return 0;

33 }

16、11行c数组排序下标范围为（0,n-1）。（ ） 否

17、12行j<n-1改为j<n，结果不会有影响。（ ） 是

18、a数组元素大小单调时，输出结果一定不为-1。 （ ）否

19、算法时间复杂度为O(nlogn)。 （ ）是

20、a数组输入为4 5 6 1 2 3，答案为：（ ） D

A．-1 B．1 C．2 D．3

21、a数组为10 6 7 8 5 1 2 4 3，答案为：（ ）。 A

A．-1 B．1 C．2 D．3

2、

1 #include<bits/stdc++.h>

2 using namespace std;

3 const int MAXN=1000000+10;

4 int n,k;

5 int a[MAXN];

6 int q[MAXN],head,tail;

7 void work() {

8 memset(q,0,sizeof(q));

9 head=tail=0;

10 for(int i=1; i<=n; i++) {

11 while(tail>head && a[i]<=a[q[tail-1]]) tail--;

12 q[tail++]=i;

13 if(q[head]+k-1<i) head++;

14 if(i>=k) printf("%d ",a[q[head]]);

15 }

16 printf("\n");

17 memset(q,0,sizeof(q));

18 head=tail=0;

19 for(int i=1; i<=n; i++) {

20 while(tail>head && a[i]>=a[q[tail-1]]) tail--;

21 q[tail++]=i;

22 if(q[head]+k-1<i) head++;

23 if(i>=k) printf("%d ",a[q[head]]);

24 }

25 }

26 int main() {

27 scanf("%d%d",&n,&k);

28 for(int i=1; i<=n; i++) scanf("%d",&a[i]);

29 work();

30 return 0;

31 }

22、程序中所用到的数据结构是栈。（ ） 否

23、两处while循环维护的q数组一定严格单调。（ ） 是

24、memset(q,0,sizeof(q))去掉后，不影响输出结果。 （ ）否

25、10-15行，每一个数进入q数组一次且离开一次。 （ ）否

26、程序运行的时间复杂度为：（ ） A

A．O(n) B．O(nlogn) C．O(n^2) D．O(n^2logn)

27、输入数据为：

8 3

1 3 -1 -3 5 3 6 7

输出结果为：（ ）。 D

A．

3 3 5 5 6 7

-1 -3 -3 3 3 3

B．

3 3 3 5 6 7

-1 -3 -3 -3 3 3

C．

-1 -3 -3 -3 3 3

3 3 5 5 6 7

D．

-1 -3 -3 -3 3 3

3 3 5 5 6 7

3、

1 #include<iostream>

2 using namespace std;

3 int a[10000];

4 int gcd(int m,int n){

5 while(n){

6 int r=m%n;m=n;n=r;

7 }

8 return m;

9 }

10 int main(){

11 int n=1000,r=202;

12 for(int i=1;i<=n-r;i++) a[i]=n-i+1;

13 for(int i=2;i<=r;i++){

14 int k=i;

15 for(int j=1;j<=n-r;j++)

16 if(gcd(k,a[j])>1){

17 int g=gcd(k,a[j]);

18 k/=g; a[j]/=g;

19 if(k==1) break;

20 }

21 }

22 int p=1,g=0;

23 for(int i=1;i<=n-r;i++){

24 p\*=a[i];

25 while(p%5==0){

26 p/=5; g++;

27 }

28 p%=5;

29 }

30 cout<<g<<endl;

31 return 0;

32 }

28、若m=2，n=0，gcd(2,0)结果为2。（ ） 是

29、19行去掉break不影响程序运行，输出结果也不变。（ ） 是

30、16行gcd(k,a[j])>1改为gcd(k,a[j])>=1，输出结果也不变。 （ ）是

31、循环结束后，a[i]两两互质。 （ ）否

32、程序运行结束a[11]等于：（ ） A

A．1 B．10 C．330 D．495

33、程序运行结束，g的答案为：（ ）。 A

A．151 B．187 C．196 D．198

完善程序

1、 一颗有根树，树上有 n 个节点（编号 1∼n ）。其中每个节点有一个苹果，每个苹果有一定的能量，现在分别选出一棵子树，要求两棵子树不能相交而且所有苹果的能量和最大。

(1<=n<=1e5,-1e4<=a[i]<=1e4)

#include <bits/stdc++.h>

using namespace std;

typedef long long ll;

const int maxn = 1e6 + 10;

const ll INF = ll(1e18);

int n;

ll a[maxn];

std::vector<int> e[maxn];

ll ans =-INF;

ll dp[maxn];

ll sum[maxn];

void dfs(int x,int fa) {

int s = 1 ;

sum[x]=a[x];

for(int i=0;i<=s-1;i++) {

int y = e[x][i];

if(y==fa) continue;

dfs(y,x);

sum[x]+=sum[y];

if( 2 ) ans=max(ans, 3 );

dp[x]= 4 ;

}

dp[x]= max(dp[x], 5 );

}

int main(){

cin>>n;

for(int i=1;i<=n;i++) cin>>a[i];

for(int i=1;i<=n-1;i++){

int x,y;

cin>>x>>y;

e[x].push\_back(y);

e[y].push\_back(x);

}

for(int i=1;i<=n;i++) dp[i]=-INF;

dfs(1,-1);

if(ans==INF) puts("Impossible");

else printf("%lld\n",ans);

return 0;

}

34、1处填（ ）。C

A．n B．1

C. e[x].size() D．x

35、2处填（ ）。C

A . ans<sum[x] B．!e[x][y]

C. dp[x]>-INF D．sum[x]>-INF

36、3处填（ ）。B

1. sum[x] B．dp[x]+dp[y]
2. dp[x] D．max(dp[x],dp[y])

37、4处填（ ）。D

1. sum[x] B．dp[x]+dp[y]
2. dp[x] D．max(dp[x],dp[y])

38、5处填（ ）。A

1. sum[x] B．dp[x]+dp[y]
2. dp[x] D．max(dp[x],dp[y])

2、 有G种颜色的宝石，有B个袋子，每个袋子里有各种颜色的宝石若干，有一个cooker

两个人轮流选一个袋子，将袋子里面的宝石全部丢到cooker里面，如果cooker里面某种颜色的宝石达到数量S就会melt成魔石。而这个操作的人就会得到魔石，而得到魔石的人可以额外获得一次选袋子的机会。现在给出G B S和每个袋子里面宝石的情况，求当2个人都选择最优的策略的时候，先手的最大收益。

注：这里的收益指先手获得的魔石数减去后手的魔石数，所以这个收益可能是负的。

输入格式：

第一行三个整数表示G、B、S（0<=B<=21, 0<=G<=8, 0<n<=10, S < 20）

第2到B+1行，每行第一个数n，表示当前袋子中宝石数量，后面跟着n个数，表示宝石颜色。

样例输入：

3 4 3

2 2 3

2 1 3

2 1 2

3 2 3 1

样例输出：

3

#include<bits/stdc++.h>

using namespace std;

typedef long long ll;

const int inf = 1<<29;

int c[22][20];//c[i][j]表示第i个袋子里面颜色为j的宝石的数量

int cooker[20];

int tmp[20];

int dp[1<<22];

int main() {

int G,B,S;

scanf("%d%d%d",&G,&B,&S);

memset(c,0,sizeof(c));

for (int i = 0; i < B; ++i) {

int n;

scanf("%d",&n);

for (int j = 0,x; j < n; ++j) {

scanf("%d",&x);

c[i][x]++;

}

}

dp[0] = 1 ;

int V = (1<<B)-1;

for (int i = 1; i <= V; ++i) {

dp[i] = -inf;

memset(cooker,0,sizeof(cooker));

for (int j = 0; j < B; ++j) {

if ( 2 ) { //0表示此处的袋子已经被拿走了，放到了cooker里面

for (int k = 1; k <= G; ++k) {

cooker[k] += c[j][k];

while (cooker[k]>=S) cooker[k] -= S;

}

}

}

for (int j = 0; j < B; ++j) {

if ( 3 ) {

for (int k = 1; k <= G; ++k) tmp[k] = cooker[k];

int get = 0;//用来保存拿j袋子的收益

for (int k = 1; k <= G; ++k) {

tmp[k] += c[j][k];

while (tmp[k] >= S) {

tmp[k] -= S;

get++;

}

}

if (get > 0) dp[i] = max(dp[i], 4 );

else dp[i] = max(dp[i], 5 );

}

}

}

printf("%d\n",dp[V]);

return 0;

}

39、1处填（ ）。A

A．0 B．1

C. inf D．-inf

40、2处填（ ）。C

A . (i&(1<<j))!=0 B．(i|(1<<j))!=0

C. (i&(1<<j))==0 D．(i|(1<<j))==0

41、3处填（ ）。A

A . (i&(1<<j))!=0 B．(i|(1<<j))!=0

C. (i&(1<<j))==0 D．(i|(1<<j))==0

42、4处填（ ）。D

A . get+dp[i&(1<<j)] B. get-dp[i&(1<<j)]

C. get-dp[i|(1<<j)] D．get+dp[i^(1<<j)]

43、5处填（ ）。B

A . dp[i&(1<<j)] B．0-dp[i^(1<<j)]

C．get-dp[i|(1<<j)] D．0-dp[i&(1<<j)]