CSP-J第一轮认证公益模拟活动

一、单选题

1、广域网的英文缩写是（A）。广域网：Wide Area Network.英文缩写WAN.通常跨接很大范围如一个国家

A. WAN　　B. LAN　　C. Internet　　D. WIFI

2、二进制数1011101与下列哪个十六进制数相同（A）。4位为一组101=5 1101=13=D

A. (5D)16　　B. (5B)16　　C. (D5)16　　D. (B5)16

3、计算机中最小的信息单位是（C）。bit指的是位

A. byte　　B. KB　　C. bit　　D. MB

4、下列哪一个循环会导致死循环（C）。k越来越小，则k<10永远成立

A、for(int k = 0;k<0;k++)

B、for(int k = 10;k>10;k--)

C. for(int k = 0;k<10;k--)

D、for(int k = 0;k>0;k++)

5、对长度位n的有序单链表，若检索每个元素的概率相等，则顺序检索到表中任一元素的平均检索长度为（B）。 计算所有元素总的检索长度，再求平均值(1+2+3...+n)/n

A、n/2　　B、(n+1)/2　　C、(n-1)/2　　D、n/4

6、把一个复杂的问题分成一个或多个的相同类似的子问题，再把子问题分解成更小的子问题...直到最后的子问题可以简单的求解，而原问题的解就是子问题解的合并，这种算法思想是（D）。这是典型的分治算法的思想

A、动态规划　　B、贪心　　C、搜索　　D、分治

7、以下程序执行完毕后，i和d的值分别为（B）i的值每次增加2，最后变成8，d的值等于2+4+6=12

```
int i, d;
d = i = 0;
for(; i <= 7; i+=2)
    d += i;
```

A、8 10　　B、8 12　　C、6 10　　D、6 12

8、G是一个非连通简单无向图，共有36条边，则该图至少有（A）个顶点。假设一个连通的无向图有n个顶点，最多有n*(n-1)/2条边，所有n*(n-1)/2=36，n=9，又因为G是非连通的，所有再外部还要加一个单独的点，所以n=10

A、10　　B、9　　C、8　　D、7

9、前缀表达式 - - * 3 5 1 + 1 2 的值是（B）。变成中缀表达式为：3*5-1-（1+2）

A、15　　B、11　　C、3　　D、13

10、一棵6节点二叉树的中序遍历为ABDGECF，先序遍历为DBACEGF,后序遍历为（B）。根据中序遍历和先序遍历先画出整颗二叉树的图，再根据图像得出后续遍历

A、DGBEFAC　　B、ABGEFCD　　C、GBEACFD　　D、ABCDEFG

11、A

解析：这道题目考察的是动态规划的数字三角形问题，C（i,j)表示a(1,1)走到a(i,j)的最大路径之和，所以考虑最后一步走到a(i,j)，取决于C（i-1，j-1)和C（i-1,j)的最大值加上a(i,j)

12、对于入栈顺序为a,b,c,d,e,f,g的序列，下列（C）不可能是合法的出栈序列。因为b一定在c之前入栈，所以b一定比c后出栈，所以C选项错误

A、a，b，c，d，e，f，g

B、a，d，c，b，e，g，f

C、a，d，b，c，g，f，e

D、g，f，e，d，c，b，a

13、下列算法中，（D）是稳定的排序算法。插入排序是稳定的

A、快速排序　　B、堆排序　　C、选择排序　　D、插入排序

14、有8本不同的书，其中3本不同的科技书，2本不同的文艺书，3本不同的体育书，将这些书竖排在书架上，则科技书连在一起文艺书也连在一起的不同排法种数为（B）种。 把科技书和文艺书分别当作一个整体和3本体育书进行排列，A（5，5）=120

科技书和文艺书再单独排列，所以答案是：120*6*2=1440

A、720　　B、1440　　C、2880　　D、3600

15、一个抽屉里有20件衬衫，其中4件是蓝的，7件是灰的，9件是红的，则应从中随意取出多少件才能保证有5件是同颜色的？D ： 最坏情况取了4件蓝，4件灰和4件红，再多取一件即可

A、5　　B、9　　C、12　　D、13

二、阅读程序

1、输入两个正整数a，b

```
#include <iostream>
#include <algorithm>
using namespace std;
long long myfun(int n, int k) {
    long long now=n, f=n;
    while(k>1) {
        if(k%2==1) {
            k--;
            f*=now;
        }
        if(k) {
            now*=now;
            k/=2;
        }
    }
    return now*f;
}
int main() {
    int a, b;
    cin >> a >> b;
    cout << myfun(a, b);
    return 0;
}
```

• 判断题

1）当输入2 7时，会输出128　　（错误） 输出256

2）当输入的a，b值超过某个范围时，程序会发生运行时错误　（错误） 不会运行错误，只是有可能结果会超出long long 上限

3）程序的时间复杂度是O(log(b)) （正确）每次循环都在原来的基础上减半

4）当输入10和18时，程序能正常得到一个正整数结果。（错误） 超出long long 上限会变成负数

• 选择题

1）第七行if中的条件换成以下哪条语句，程序的功能不变。C 判断奇偶性实际上就是判断这个数字的二进制形式的最后一位是0还是1

A、k|1 B、!k|1 C、k&1 D、!k&1

2、输入n，随后输入n行指令

```
# include <iostream>
using namespace std;
const int N=100010;
int ne[N],e[N];
int hh;
int n;
char op;
int k,x;
int idx;
void init(){
    hh=-1;
    idx=0;
}
void add_head(int x){
    e[idx]=x;
    ne[idx]=hh;
    hh=idx;
    idx++;
}
void insert(int k,int x){
    e[idx]=x;
    ne[idx]=ne[k];
    ne[k]=idx;
    idx++;
}
void remove(int k){
    if(ne[k]!=-1){
        ne[k]=ne[ne[k]];
    }
}
int main(){
    cin>>n;
    init();
```

```
    while(n--){
        cin>>op;
        if(op=='H'){
            cin>>x;
            add_head(x);
        }
        else if(op=='D'){
            cin>>k;
            if(!k) hh=ne[hh];
            else remove(k-1);
        }
        else if(op=='I'){
            cin>>k>>x;
            insert(k-1,x);
        }
    }
    for(int i=hh;i!=-1;i=ne[i]){
        cout<<e[i]<<" ";
    }
    return 0;
}
```

• 判断题

1）当输入 H x 时，程序会在链表的开头插入 x 元素。 （正确）

2）当输入 D 0 时，程序会删除当前头节点。（正确）

3）当输入执行insert或add_head操作N次后，再执行1次remove操作，即可腾出空间继续插入元素。（错误） remove操作通过改变next域进行删除，但是数组空间并没有清空

• 选择题

1）当输入为：

5

H 1

H 2

H 3

D 0

D 1

输出为：A 按照顺序先在头节点插入123链表变成321，再删除头节点变成2，1,D 1在这里没有用，因为头节点已经被删掉了

A、2 1　　B、3 2　　C、1　　D、2

2）当输入为：

5

H 1

I 1 3

I 2 2

D 2

D 1

输出为：C　链表首先经过插入变成 1 3 2，然后删除2号节点变成1 3，再删除1号节点变成1

A、3　　B、1 2　　C、1　　D、1 3

3）在这个简单的链表中，插入与查询的复杂度分别为（假设链表元素有n个）：C

A、O(n) O(1)　B、O(n) O(n)　　C、O(1) O(n)　　D、O(1) O(1)

3、

```
# include <iostream>
# include <string>
using namespace std;
bool check(string s,int start,int end){
    while(start<end){
        if(s[start]!=s[end]){
            return false;
        }
        start++;
        end--;
    }
    return true;
}
int f[100010];
int mincut(string s){
    int m=s.length();
    if(m==0){
        return 0;
    }
    for(int i=0;i<=m;i++){
```

```
            f[i]=i-1;
        }
        for(int i=1;i<=m;i++){
            for(int j=0;j<i;j++){
                if(check(s,j,i-1)){
                    f[i]=min(f[i],f[j]+1);
                }
            }
        }
        return f[m];
}
int main(){
        string s;
        cin>>s;
        cout<<endl<<mincut(s);
}
```

• 判断题

1）check函数在判断整个字符串s是否为一个回文串。（错误）判断s的一部分是否为回文串

2）输入 aaabaaa 程序会输出0。（正确）

3）把mincut函数中的

```
for(int i = 0;i <= m;i++){
    f[i] = i-1;
}
```

修改为

```
for(int i = 1;i <= m;i++){
    f[i] = i-1;
}
```

修改前后输入aaa，程序得到的结果不会有变化。（错误）会从0变成1

• 选择题

1）输入 aababccba ，输出为：A 这个代码是在通过动态规划求将字符串s划分为若干个回文子串，使得划分所需的“切割”次数最少：aa | b | abccba

A、2　　B、3　　C、4　　D、5

2）假如输入的字符串长度为n，程序的复杂度为：B

A、O(n^2)　　B、O(n^3)　　C、O(n^2 * log(n))　　D、O(n)

四、完善程序

（一）、全排列问题，输出n (n<10)，输出n的全排列。

如输入：

3

输出：

1 2 3

1 3 2

2 1 3

2 3 1

3 1 2

3 2 1

使用深度优先搜索解决问题。

请补全程序。

```
#include <iostream>
using namespace std;
int a[15], n;
bool vis[15];
void print() {
    for(int i = 1; i <= n; i++)
        cout << a[i] << " ";
    cout << endl;
}
void dfs(int step) {
    if(step==n+1) {
        print();
        return;
    }
    for(int i = 1; i <= n; i++) {
        ① ;
        a[step] = i;
        ②
        dfs( ③ );
        ④ ;
    }
}
int main() {
    cin >> n;
    dfs( ⑤ );
    return 0;
}
```

1)①处应填(B) 如果这个元素被访问过就continue

A、if(vis[i]) break

B、if(vis[i]) continue

C、if(!vis[i]) break

D、if(!vis[i]) continue

2)②处应填(C) 将这个元素标记访问

A、vis[i] = 0

B、a[i]=step

C、vis[i]=1

D、a[i]=step+1

3)③处应填(A) 搜索下一个位置应该填什么数字

A、step+1
B、step
C、i+1
D、i

4)④处应填(A) 回溯，将标记取消
A、vis[i] = 0
B、a[i]=step
C、vis[i]=1
D、a[i]=step+1

5)⑤处应填(B) 从第一个位置开始搜索
A、0
B、1
C、n
D、n-1

（二）、使用SPFA解决单源最短路问题。
输入一张有向图，n个点，m条边，每条边输入x,y,w，表示x到y有一条权值为w的边。

```
#include <queue>
using namespace std;
const int maxn=10005;
struct edge{
    int v,w;
};
vector<edge> e[maxn];
int dis[maxn],vis[maxn];
queue<int>q;
int n,m;
bool spfa(int s){
    for(int i=1;i<=n;i++) dis[i]=1e9;
    ① ;
    while(!q.empty()){
        int u=q.front();
        q.pop(),vis[u]=0;
        for(int i=0;i<e[u].size();i++){

            int v = e[u][i].v, w = e[u][i].w;
            if( ② ) {
                ③ ;
                if(!vis[v]) ④ ;
            }
        }
    }
    return true;
}
int main() {
    cin >> n >> m;
    while(m--) {
        int x, y, w;
        cin >> x >> y >> w;
```

```
    ⑤ ;
}
int s;
cin >> s;
spfa(s);
for(int i = 1; i <= n; i++)
    if(dis[i]!= ⑥ ) cout << dis[i] << endl;
    else cout << "NO PATH" << endl;
return 0;
}
```

1)①处应填(C) 初始化s到s的距离为0，s这个点标记已经被访问了

A、dis[s] = 0, vis[s] = 0
B、dis[s] = -1, vis[s] = 1
C、dis[s] = 0, vis[s] = 1
D、dis[s] = -1, vis[s] = 0

2)②处应填(B) 如果更新后距离比原来的最短距离小

A、dis[v] < dis[u] + w
B、dis[v] > dis[u] + w
C、dis[u] < dis[v] + w
D、dis[u] > dis[v] + w

3)③处应填(D) 更新距离

A、dis[u] = dis[v] - w
B、dis[v] = dis[u] - w
C、dis[u] = dis[v] + w
D、dis[v] = dis[u] + w

4)④处应填(A) 将更新后的点加入到队列当中，并标记

A、q.push(v), vis[v] = 1
B、q.push(v)
C、q.push(u), vis[v] = 1
D、q.push(u)

5)⑤处应填(B) 将x-y这条权值为w的边储存到邻接表当中

A、e[x].push_back((edge){y, w}), e[y].push_back((edge){x, w})
B、e[x].push_back((edge){y, w})
C、e[x].push_back((edge){w, x})
D、e[y].push_back((edge){x, w})

6)⑥处应填(B) 如果最短路径还是无穷大，则表示不能到达

A、1
B、1e9
C、0
D、-1

详情咨询请添加下方高老师微信