2020 CCF非专业级别软件能力认证第一轮

(CSP-S)提高级 C++语言试题

**认证时间：2020年10月11** **日09** **:30~11** **:30**

**分数组成：单项选择题15题共：30分**

**一、单项选择题**

**1.**请选出以下最大的数 ( )

A. (550)10 B. (777)8 C. 210 D. (22F)16

解析：答案C A=550B=7\*8^2+7\*8^1+7\*8^0=551 C=1024,D=2\*16^2+2\*16^1+15=559

**2** **.**操作系统的功能是( )。

A.负责外设与主机之间的信息交换

B.控制和管理计算机系统的各种硬件和软件资源的使用

C.负责诊断机器的故障

D.将源程序编译成目标程序

解析：答案 B 操作系统是计算机系统的关键组成部分，负责管理与配置内存、决定系统 资源供需的优先次序、控制输入与输出设备、操作网络与管理文件系统等基本任务。

**3.** 现有一段8分钟的视频文件，它的播放速度是每秒24帧图像，每帧图像是 一幅分辨率

为2048 × 1024像素 的32位真彩色 图像 。请 问要存储这段原始无 压缩视频 ， 需 要 多大 的存储 空 间? ( )。

A. 30G B. 90G C. 150G D. 450G

解析：信息存储单位答案 B

字节 (Byte)=8 bit(位) 32/8 =4

一个像素是32位真彩色，也就一个像素占4个字节 1M=1024\*1024B

1G=1024\*1024\*1024B 8分钟=8\*60秒

8\*60\*24\*4\*2048\*1024/(1024\*1024\*1024)=90G

**4.** 今有一空栈S,对下列待进栈的数据元素序列a,b,c,d,e,f依次进行：进栈，进栈，出

栈，进栈，进栈，出栈的操作，则此操作完成后，栈底元素为 ( )。

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

解析：**答案** **B** **只要按照进栈出栈顺序模拟一下即可**

**5.**将 (2,7, 10, 18)分别存储到某个地址区间为**0~** **10** 的哈希表 中，如果

哈希函数h(x)=（） , 将**不会**产生冲突，其中**a** **mod** **b**表示a 除以b的余数。

A. x²mod 11

B. 2x mod 11

C. x mod 11

D. [x/2]mod 11,其中[x/2] 表示x/2下取整

问题解析：答案 D

A: x² mod 11 2->4 7->5 10->1 18->5 会产生冲突

B: 2x mod 11 2->4 7->3 10->9 18->3 会有冲突

C: x mod 11 2->2 7->4 10->10 18->7 会有冲突

D: [x/2]mod 11 2->1 7->3 10->5 18->7 没有冲突

**6.** 下列哪些 问题**不能**用贪心法精确求解? ( )

A. 霍夫曼编码问题 B.0-1 背包问题

C. 最小生成树问题 D.单源最短路径问题

解析：**答案** **B** 贪心策略也可以求解完全背包，但不能求解0-1背包问题

**7.** 具有n个顶点，e条边的图采用邻接表存储结构，进行深度优先遍历运算的时间复杂 度为( )。

A. θ (n+e) B. θ (n²) C. θ (e²) D. θ (n)

解析：答案A [存储结构用的邻接表](https://so.csdn.net/so/search?q=%E9%82%BB%E6%8E%A5%E8%A1%A8&spm=1001.2101.3001.7020)，那么，就是依次访问每一结点和边，即 θ (n+e)

**8.** 二分图是指能将顶点划分成两个部分，每一部分内的顶点间没有边相连的简单无向图

。那么,24个顶点的二分图**至多**有 ( )条边。

A.**144** B.100 C.**48** D.**122**

解析：答案A 点集分为 24/2=12 的两部分，每部分的一个点向另一部分的每一个点连一 条边，12\*12=144 条

**9.** 广度优先搜索时，一定需要用到的数据结构是( )。

A. 栈 B.二叉树 C.队列 D.哈希表

解析：答案C 这是个基础题，BFS需要借助队列来实现逐层的访问，DFS用递归的形式来实 现，需要栈。

10.一个班学生分组做游戏，如果每组三人就多两人，每组五人就多三人，每组七人就多四

人，问这个班的学生人数n 在以下哪个区间?已知n<60。( )。

A.30<n<40 B.40<n<50 C.50<n<60 D.20<n<30

解析：答案C n<60时满足n%3=2、n%5=3和n%7=4数只有53,选择 C

**11.**小 明想通过走楼梯来锻炼身体 ，假设从第 1层走到第**2**层消耗 **10**卡热量 ，接着从

第2层走到 第 3 层消 耗 2 0卡 热 量 ，再从第**3** 层走到第层消耗**30**卡热量，依此类推，从

第k 层 走 到 第 k+1层消耗10k卡热 量(k>1)。如果小明想从 1层开始 ，通过连续 向

上爬楼梯消耗1000卡热量 ，**至少**要爬到第几层 楼 ? ( )。

A. 4 B.16 C.15 D.13

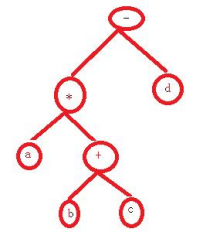
解析：答案C从K层走到K+1层消耗热量是10K从1层走到K+1层消耗的热量是：10+20+…

+10K=10(1+2+…+K)=10\*k\*(K+1)/2=5\*K\*(K+1)5\*K\*(K+1)>=1000K=14,要爬到K+1层，是第 15层

**12.** 表 达 式a\*(b+c)-d的后缀表达形式为( )。

A.abc\*+d- B.-+\*abcd C.abcd\*+- D.abc+\*d-

解析：**答案D** **画出该表达式树，后序遍历表达式树则为后缀表达式**



**13.** **排列组合**

从一个4×4的棋盘中选取不在同一行也不在同一列上的两个方格，共有( )

种方法。

A.60 B.72 C.86 D.64

|  |  |  |  |
| --- | --- | --- | --- |
| 1 | 2 | 3 | **4** |
| 5 | 6 | 7 | **8** |
| 9 | 10 | 11 | **12** |
| 13 | 14 | 15 | **16** |

4行4列共有16个格，

选第一个格有16种选法，再选第二个格子， 第二个格子有，9种选法

这样的话，会出现重复的情况比如：第一次有可能选6,也有可能选11

选 1 1,6 与6, 1 1 是 同 一种方法答案应为： C(16,1)\*C(9,1)/2=72 种方法

**答案** **B**

**14.**对一个n 个顶点、m 条边的带权有向简单图用Dijkstra 算法计算单源最短路时，如果

不使用堆或其它优先队列进行优化，则其时间复杂度为( )。

A. θ ((m+n²)logn) B. θ (mn+n³)

C. θ ((m+n)logn) D. θ (n²)

解析：答案D Dijkstra算法每次寻找最短路径的时间复杂度O(n),整个算法的时间复杂度 为O(n2).带堆优化的Dijkstra算法每次寻找最短路径时采用堆排序，时间复杂度O(logn) 。整个算法的时间复杂度优化为O(n\*logn).

**15.** 1948年( )将热力学中的熵引入信息通信领域，标志着信息论研究的开端。

A. 欧拉 (Leonhard Euler) B.冯 ·诺伊曼 (John von Neumann)

C. 克劳德 ·香农 (Claude Shannon) D.图灵 (Alan Turing)

解析：答案 C

二 、阅读程序阅读程序题：3题(判断、选择) 共40分

阅读程序(程序输入不超过数组或字符串定义的范围；判断题正确填 v,错误填x;除特殊说明外， 判断题1.5分，选择题3分，共计40分)

**1.** **阅读程序1**

1. #include<iostream>

2. using namespace std;

3.

4. int n;

5. int d[1000]; 6.

7. int main(){

8. c in >>n;

9. for(inti=0;i<n;i++){

10. c in>>d[i];

11. }

12. int ans =-1;

13. for(inti=0;i<n;i++)

14. for(int j=0;j<n;j++)

15. if(d[i]<d[j])

16. ans=max(ans,d[i]+d[j]-(d[i]&d[j]));

17. cout<<ans;

18. return 0;

19. }

假设输入的 n 和 d[i] 都是不超过10000 的正整数。 判断题：

1)n必须小于1000,否则程序可能会发生错误。 (错误)

解析：答案错误 当n=1000时也可以

2)输 出一定大于等于0 。 (错误)

解析： 答案 :错误 当n= 1 d[0]= 1输 出-1

3)若将第13行的 “j=0 ” 改为“j=i+1 ”, 程序输出可能会改变。 ( 正 确 )

解析：答案正确 当j=i+1时 循环比较次数会发生变化，结果也有可能发生变化。

4 ) 将 第 1 4 行 的“d[i]<d[j] ”改为“d[i]!=d[j] ”,程序输出不会改变。 (正确)

解析：答案正确 i,j都判断一次di<dj ，就相当于判断di !=dj, 因为在第15行求最大值， 对结果没有影响。

5)若输入的n为100,且输出为127,则输入的 d[i] 中不可能有 (C)

A.127 B.126 C.128 D.125

解析：答案n=2 d[0]=128 d[1]=0 会输出128

6 ) 若 输 出 的 数 大 于 0 , 则 下 面 说 法 正 确 的 是 ( )

A . 若 输 出 为 偶 数 ， 则 输 入 的 d [ i ] 中 最 多 有 两 个 偶 数 B . 若 输 出 为 奇 数 ， 则 输 入 的 d [ i ] 中 至 少 有 两 个 奇 数 C . 若 输 出 为 偶 数 ， 则 输 入 的 d [ i ] 中 至 少 有 两 个 偶 数 D . 若 输 出 为 奇 数 ， 则 输 入 的 d [ i ] 中 最 多 有 两 个 奇 数

解 析 ： 答 案 C 举 反 例 n = 3 , d 序 列 为 2 , 4 , 6 , 输 出 为 6 n = 3 , d 序 列 为 1 , 3 , 5 , 输 出 为 7

n = 2 , d 序 列 为 1 , 2 , 输 出 为 3

**2.** **阅读程序2**

1. #include <iostream>

2. #include <cstdlib>

|  |  |  |
| --- | --- | --- |
| 3. using namespace std; | | |
| 4. | | |
| 5. | int | n; |
| 6. | int | d[ 10000]; |
| 7. | | |
| 8. int find(int L,int R,int k){ | | |
| 9. int x=rand()%(R-L+1)+L; | | |
| 10. | swap(d[L],d[x]); | |
| 11. | int a=L+1,b=R; | |
| 12. | while (a<b){ | |
| 13. | while (a<b&&d[a]<d[L]) | |
| 14. | ++a: | |
| 15. | while (a<b &&d[b]>=d[L]) | |
| 16. | --b; | |
| 17. | swap(d[a],d[b]); | |
| 18. | } | |
| 19. | if (d[a]<d[L]) | |

20. ++a:

I 21. if(a-L==k)

22. return d[L]

23. if(a-L<k)

24. return find(a,R,k-(a-L))

25. returnfind(L+1,a-1,k);

26. }

27.

28. int main(){

29. intk;

30. c in>>n;

31. c in>>k;

32. for (int i=0;i<n;++i)

33. c in>>d[i];

34. cout<<find(0,n-1,k);

35. return0:

**36.** }

假设输入的 n,k和d[i]都是不超过10000的正整数，且k不超过n,并假设 rand()函

数产生的是均匀的随机数。 判断题：

1 ) 第 9 行 的 “x” 的数值范围是 L+1 到 R, 即 [L+1,R] 。 ( 错 误 )

解析：答案错误 如果L-R+1>=0，x%(R-L+1)的范围是[0,R-L],范围变为[L,R]，所 以答案错误。

2)将第19行的“d[a] ”改为“d[b] ”, 程序不会发生运行错误。 ()

解析：解析：判断运行错误，主要看是否有越界、除0等情况发生，这里没有除法， 为b=R,R<=n,n<=10000所以数组也不会越界。程序不会发生运行错误。

3)当输入的 d[i] 是严格单调递增序列时，第17 行的“**swap** **”**的平均执行次 数 是 ( )

A. θ (nlogn) B. θ (n) C. θ (logn) D. θ (n2)

解析：答案应为log2n，四个选项均为错误，比赛时任选一个选项都被判对。因为这是严 格递增数列，所以前面++a和--b每次都会执行R-L-1次后直接跳出大循环。递归最多执行 logn次 ，R-L-1平均为logn,则swap平均执行次数为log2n

理想的情况是，每次划分所选择的中间数恰好将当前序列几乎等分，经过log2n趟划 分，便可得到长度为1的子表。这样，整个算法的时间复杂度为O(nlog2n)。

最坏的情况是，每次所选的中间数是当前序列中的最大或最小元素，这使得每次划分 所得的子表 中一个为空表，另一子表的长度为原表的长度-1。这样，长度为n 的数据表 的快速排序需要经过n 趟划分，使得整个排序算法的时间复杂度为O(n²)

4)当输入的 d[i] 是严格单调递减序列时，第17 行的“swap ”平均执行次 数 是 ( )

A. θ (n²) B. θ (n) C. θ (nlogn) D. θ (logn)

解析：答案B θ (n/2)+n/2=n/2+n/4+n/8+………= θ (n)

5)若输入的 d[i] 为 i,此程序①平均的时间复杂度和②最坏情况下的时间复杂度分别 是 (A)

A. θ (n), θ (n²) B. θ (n), θ (nlogn)

C. θ (nlogn), θ (n²) D. θ (nlogn), θ (nlogn)

解析：答案A 平均情况为每次递归都是均匀分布，恰好分成两半，时间复杂度为θ (n)最坏情况为每 次递归都是左边一个，右边n-1个，这样会跑(n-1)+(n-2)+…+1=n^2次， θ (n)=n^2

6)若输入的 d[i]都为同一个数，此程序的平均时间复杂度是 (D)

A. θ (n) B. θ (logn) C(nlogn)D θ (n²)

**解析：答案D** **每次程序都会扫一遍整个数组的后n-1个数，时间复杂度为θ** **(n2** )

**2.** **阅读程序**

1. #include<iostream>

2. #include<queue>

3. usingnamespacestd; 4.

5. constintmaxl=2000000000; 6.

7. classMap{

8. struct item{

9. stringkey;intvalue;

10. }d[maxl];

11. intcnt;

12. public :

13. intfind(stringx){

14. for(int i=0;i<cnt;++i)

15. if(d[i].key==x)

16. return d[i].value;

17. return-1;

18. }

19. staticintend(){return-1;}

20. void insert(string k,int v){

21. d[cnt].key=k;d[cnt++].value=v;

22. }

23. }s[2]; 24.

25. classQueue{

26. string q[maxl];

27. inthead,tail;

28. public :

29. voidpop(){++head;}

30. string front(){return q[head +1];}

31. bool empty(){return head ==tail;}

32. void push(string x){q[++tail]=x;}

33. }q[2]; 34.

35. stringst0,st1;

36. int m;

37.

38. string LtoR(string s,int L,int R){

39. stringt=s;

40. char tmp=t[L];

41. for(inti=L;i<R;++i)

42. t[i]=t[i+1];

43. t[R]=tmp;

44. return t;

45. } 46.

47. stringRtoL(strings,intL,intR){

48. string t=s;

49. chartmp=t[R];

50. for(int i=R;i>L;--i)

51. t[i]=t[i-1];

52. t[L]=tmp;

53. returnt;

54. }

55.

56. bool check(string st,int p,int step){

57. if(s[p].find(st)!=s[p].end())

58. return false;

59. ++step;

60. if(s[p^1].find(st)==s[p].end()){

61. s[p].insert(st,step);

62. q[p].push(st);

63. returnfalse;

64. }

65. cout<<s[p^1].find(st)+step<<endl;

66. return true;

67. } 68.

69. intmain(){

70. c in >>st0>>st1;

71. intlen=st0.length();

72. if(len !=st1.length()){

73. cout<<-1<<endl;

74. return 0;

75. }

76. if(st0==st1){

77. cout<<0<<endl;

78. return 0;

79. }

80. c in >>m;

I 81. s[0].insert(st0,0);s[1].insert(st1,0);

82. q[0].push(st0);q[1].push(st1);

83. for(intp=0;

84. !(q[0].empty()&&q[1].empty());

85. p^=1){

86. string st=q[p].front();q[p].pop();

87. intstep=s[p].find(st);

88. if((p==0 &&

89. (check(LtoR(st,m,len-1),p,step)||

90. check(RtoL(st,0,m),p,step)))

91.  ||

92. (p==1&&

93. (check(LtoR(st,0,m),p,step)||

94. check(RtoL(st,m,len-1),p,step))))

95. return0;

96. }

97. cout<<-1<<endl;

98. return 0;

99. }

判断题：

1)判断：输出可能为0。 (正确)

解析：答案正确 输入的两字符串相等会输出0。

2)若输入的两个字符串长度均为101时 ，则m=0时的输出与m=100时的输出是 一 样的。 ( 错误 )

解析：答案错误 m=0或者100时分别做的是整体循坏左移或者循环右移，不同的操作输出可不一 样。

3)若两个字符串的长度均为 n, 则最坏情况下，此程序的时间复杂度为O(n!)。 ( 错 误 )

解析：答案错误 最坏情况下字符串的排列O(n!),查询也是O(n!),判断是否相等时O(n),复杂度为 O(n\*(n!)2)

4）若输入的第一个字符串长度由100个不同的字符构成，第二个字符串是第一个字符串的倒序， 输入的m为0,则输出为 (D)。

A.49 B.50 C.100 D.-1

5)已知当输入为“0123\n3210\n1”时输出为4，当输入为“012345\n543210\n1”时输出为14，当输入为“01234567\n76543210\n1”时输出为28，则当输入为“0123456789ab\nba9876543210\n1”输出位（）。其中“\n”为换行符。

A.56 B.84 C.102 D.68

解析：答案D

解析：



发现字符串长度为4时答案为4，长度为6时答案为14，长度为8时答案 为28，差分别为10,14。于是猜测接下来的两个差分别为18,22，于是长度为12时答案为

28+18+22=68。或者考虑这是一个二次函数，代入求解解析式再计算答案。

6) 若两个字符串的长度均为 n, 且 0<m<n-10<m<n-1, 且两个字符串的构成相同(即任何一 个字符在两个字符串中出现的次数均相同),则下列说法正确的是()。提示：考虑输入与输 出有多少对字符前后顺序不一样。

A.若n、m均为奇数，则输出可能小于0

B.若n、m均为偶数，则输出可能小于0

C.若n为奇数、m为偶数，则输出可能小于0 D.若n为偶数、m为奇数，则输出可能小于0

**解析：答案(C)** **如果移位的长度是奇数，不会改变逆序对的奇偶性，如123移位为231，奇偶性** **不变，如果是偶数1234移位为2341，逆序对个数奇偶性改变。所以选C**

**三、完善程序**完善程序题：2题 (选择) 共 3 0 分

**1.完善程序** **1**

(分数背包)小S有 n块蛋糕，编号从1到n。第i块蛋糕的价值是wi,体积是vi。他有一个 大小为 B 的盒子来装这些蛋糕，也就是说装入盒子的蛋糕的体积总和不能超过B。

他打算选择一些蛋糕装入盒子，他希望盒子里装的蛋糕的价值之和尽量大。

为了使盒子里的蛋糕价值之和更大，他可以任意切割蛋糕。具体来说，他可以选择一个 a(0<a<1),并将一块价值是w,体积为v的蛋糕切割成两块，其中一块的价值是αw, 体积

是av,另一块的价值是(1-a)w,体积是(1-a)v 。他可以重复无限次切割操作。 现要求编程输出最大可能的价值，以分数的形式输出。

比如 n=3,B=8,三块蛋糕的价值分别是4、4、2,体积分别是5、3、2。

那么最优的方法就是将体积为5的蛋糕切成两份，一份体积是3,价值是2.4,另一份体积 是2,价值是1.6,然后把体积是3的那部分和后两块蛋糕打包进盒子。最优的价值之和是 8.4,故程序输出42/5。 输入的数据范围为：1≤ n≤1000,1≤B≤10⁵ ,1≤w;,v; ≤100。

提示：将所有的蛋糕按照性价比w;/v; 从大到小排序后进行贪心选择。 试补全程序。

1. #include <cstdio>

2. using namespace std;

3.

4. const int maxn = 1005;

5.

6. int n,B,w[maxn],v[maxn];

7.

8. int gcd(int u,int v){

9. if(v==0)

10. return u;

11. returngcd(v,u%v);

12. }

13.

14. void print(int w,int v){

15. intd=gcd(w,v);

16. w=w/d;

17. v=v/d;

18. if (v ==1)

19. printf("%d\n",w);

20. else

21. printf("%d/%d\n",w,v);

22. }

23.

24. void swap(int &x,int &y){

25. intt=x;x=y;y=t;

26. }

27.

28. int main() {

29. scanf("%d%d",&n,&B);

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

31. scanf("%d%d",&w[i],&v[i]);

32. }

33. for(inti=1;i<n;i++)

34. for(int j=1;j<n;j++)

35. if(① ){

36. swap(w[j],w[j+1]);

37. swap(v[j],v[j+1]);

38. }

39. intcurV,curW;

40. if (② ){

41. ③

42. } else {

43. print(B\*w[1],v[1]);

44. return 0;

45. } 46.

47. for(inti=2;i<=n;i++)

48. if(curV+v[i]<=B){

49. curV+=v[i];

50. curW+=v[i];

51. }else{

52. print(④ );

53. return0;

54. }

55. print(⑤ );

56. return 0;

57. }

①处 应 填 ( )

A.w[j]/v[i]<w[j+1]/v[j+1] B.w[j]/v[j]>w[j+1]/v[j+1]

C.v[j]\*w[j+1]<v[j+1]\*w[j]

D.w[j]\*v[j+1]<w[j+1]\*v[i]

解析：答案D 性价比更高的在前面，否则进行交换，同时为避免整数除法可能出现误差， 将除法转为乘法。

②处 应 填 ( )

A.w[1]<=B B.V[1]<=F C.w[1]>=B D.v[1]>=B

**解析：答案B** **性价比最高的蛋糕体积是否装满盒子**

③处应填( )

A.print(v[1],w[1]) ;return 0 ;

B.curV=0 ;curW=0 ;

C.print(w[1],v[1]) ;return 0 ; D.curV = v [1] ;curW= w[1] ;

**解析：答案D** **下面for从2开始，需要先初始化1。**

④处应填()

A.curW\*v[i]+curV\*w[i],v[i]

B. (curW-w[i])\*v[i]+(B-curV)\*w[i],v[i]

C.curW+v[i],w[i]

D.curW\*v[i]+(B-curV)\*w[i],v[i]

**解析：答案D** **print输出的是w/v，前curW是可以完整选走的部分，所以分子必定包含curW**

**\*** **v[i]，排除** **B** **和** **C** **，剩下的部分是B-curV**

⑤处应填()

A.curW,curV B.curW,1 C.curV,curW D.curV,1

**解析：答案B** **此时没有任何蛋糕被切掉，输出curW，则分母应该选B**

3.完 善 程 序 2

(最优子序列)取m=16, 给出长度为n的整数序列a1,a2, …,an(O≤ai≤2m)。对于 一 个二进制

数x, 定义其分值w(x)为x+popcnt(x),其中popcnt(x)表示x二进制表示中1的个数。对于一个 子序列b1,b2, …,bk,定义其子序列分值S为 w(b1 ⊕b2)+w(b2 ⊕b3)+w(b3 ⊕b4)+…w(bk-1 ⊕bk)。其

中⊕表示按位异或。对于空子序列，规定其子序列分值为0。求一个子序列似的其子序列的 分值最大，输出这个最大值。

输入第一行包含一个整数n(1≤n≤40000) 。接下来一行包含n个整数a1,a2,…,an。

提示：考虑优化朴素的动态规划算法，将前m/2位和后m/2位分开计算。

Max[x][y]表示当前的子序列下一个位置的高8位是x、最后一个位置的低8位是y时的最大价 值。

试补全程序。

1. #include <iostream>

2.

3. using

std;

nanespace

4.

5. typedef long long LL;

6.

7. const int MAXN =40000,H=16,B=H>1,MS=(1<<B)-1;

8. const LL INF =1000000000000000LL;

9. LL Max[MS+4][MS+4];

10.

11. int w(int x)

12.

{

13. int s = x;

14. while (x)

15. {

16. ① ;

17. s++;

18. }

19. return s;

20. }

21.

22. void to\_max(LL &x,LL y)

23. {

24. if (x<y)

25. x=y;

26. }

27.

28. int main()

{

29.

30. int n;

31. LL ans=0;

32. c in >>n;

33. for(int x=0;x<=MS;x++)

34. for (int y=0;y<=MS;y++)

35. Max[x][y]=-INF;

36. for(int i=1;i<=n;i++)

37. {

38. LL a;

39. c in >>a;

40. int x=②,y =a &MS;

41.

LL

v=③;

42. for (int z=0;z<=MS;z++)

43. to\_max(v,④ );

44. for(int z=0;z<=MS;z++)

45. ⑤;

46. to\_max(ans,v);

47. }

48. cout<<ans<<endl;

49. return0;

50. }

①处应填()

A.x>>=1 B.x^=x&(x^(x+1)) C.x-=x|-x D.x^=x&(x^(x-1))

**解析：答案D。等价于求lowbit。**

②处应填 ()

A. (a&MS)<<B B.a>>B C.a&(1<<B) D.a&(MS<<B)

**解析：答案B** **数的后八位，** **x** **则是去这个数的高八位，就是** **a>>B。**

③处应填( )

A.-INF B.Max[y][x] C.0 D.Max[x][y]

**解析：答案C** **方程赋初值，Max[x][y]表示当前的子序列下一个位置的高8位是x、最后一个** **位置的低** **8** **位是** **y** **时的最大价值。v=max{Max[x][y]+w(y^z)},此时赋初值应该为0**

④处应填( )

A.Max[x][z]+ w(y ^z) B.Max[x][z]+ w(a ^z)

C.Max[x][z]+ w( x ^( z<< B))

D.Max[x][z]+ w( x ^z)

**解析：答案A** **最大值取Max[** **x][** **z** **]和将y** **^** **z** **进** **行w运算的和**

⑤处应填( )

A.to\_max(Max[y] [z],v+w(a^ (z<< B))) B.to\_max(Max[z] [y],v+w((x^z)<<B)) C.to\_max(Max[z] [y],v+w(a^ (z<<B))) D.to\_max(Max[x] [z],v + w(y^z))

**解析：答案** **B** **结合第4题** **，这次是** **固定y** **，变换z** **，故排除A和D** **，变换x** **的值也时时** **更新** **。和上一题一样** **，x^z是补充z剩下的** **11** **的贡献** **。下一个位置** **的高8位是x** **，所** **以x需要恢复到高位** **的贡献** **，再与v相加。**