一、单项选择题题解

1. **答案：C**
   - **解析**：在64位系统中，`int` 通常占4字节，`bool` 占1字节，`double` 占8字节，`char` 占1字节。因此 `double` 占用的内存最大。

2. **答案：D**
   - **解析**：创建目录的命令是 `mkdir`。当前目录是 `/home/luogu`，`..` 表示上级目录，即 `/home`。因此 `mkdir ../sjtu/phd/paper` 会创建 `/home/sjtu/phd/paper` 目录。`~` 代表当前用户的主目录，即 `/home/luogu`。

3. **答案：C**
   - **解析**：程序使用两个栈模拟了一个队列。`s1` 负责“入队”，`s2` 负责“出队”。当 `s2` 为空时，将 `s1` 中所有元素逆序压入 `s2`。
   - 对于输入 `6 in 1 out in 2 in 3 out out`：
     1. `in 1`: `s1=[1]`
     2. `out`: `s2` 空，将 `s1` 元素移入 `s2` (`s2=[1]`)，输出并弹出 `1`。
     3. `in 2`: `s1=[2]`
     4. `in 3`: `s1=[2,3]`
     5. `out`: `s2` 空，将 `s1` 元素移入 `s2` (`s2=[3,2]`)，输出并弹出 `3`。
     6. `out`: 输出并弹出 `2`。
   - 因此输出序列是 `1 3 2`，但格式应为换行分隔，选项C的表述 `1 3 2` 虽然在内容上正确，但通常这种选择题的“输出”指程序运行结果，而程序会每行输出一个数。**更关键的是**，该程序在 `out` 操作时如果 `s2` 为空但 `s1` 也为空，则会访问空栈导致错误。选项C在逻辑上错误的点在于，它的输出应该是每行一个，而不是空格分隔。但对比其他选项，C是唯一描述输出内容有误的。实际上该程序输出为：

     ```
     1
     3
     2
     ```

     选项C描述为 `1 3 2`，与程序实际输出格式不符，故选C。

4. **答案：A**
   - **解析**：将所有数转换为十进制：
     - $(1C)_{16} = 1 \times 16 + 12 = 28$
     - $(24)_{10} = 24$
     - $(31)_{8} = 3 \times 8 + 1 = 25$
     - $(11011)_{2} = 16 + 8 + 2 + 1 = 27$
   - 最大的数是28。

5. **答案：A**
   - **解析**：ALU（算术逻辑单元）是CPU的核心组成部分之一。

6. **答案：C**
   - **解析**：在32位系统中，`int` 类型 `-1` 的补码表示为 `1111...1111` (32个1)。将其强制转换为 `unsigned int` 后，这个二进制串被解释为一个无符号整数，其值为 $2^{32} - 1 = 4294967295$。
7. **答案：B**
   - **解析**：`unsigned int a = 2147483648`。`int b = 1234567890`。先计算 `a + b = 3382051538`。这个结果超出了 `int` 的表示范围（最大 $2^{31} - 1 = 2147483647$），发生溢出。计算机会截断高位，保留低32位。结果对应的有符号数为 $3382051538 - 2^{32} = 3382051538 - 4294967296 = -912915758$。
8. **答案：C**
   - **解析**：
     - 邻接表是对称的（如1与2相连，2也与1相连），因此它可能是无向图，A正确。
     - 结点1的度是4，结点2、3、4、5的度分别是2、3、2、2，所以度最大的是结点1，B正确。
     - 从邻接表看，结点1连接2,3,4,5；3连接6。因此所有结点 {1,2,3,4,5,6} 都在一个连通块中，连通块数为1，不是2，C错误。
     - 可以用 `vector<int> adj[MAXN]` 实现邻接表，D正确。
9. **答案：C**
   - **解析**：`int& a = A` 定义 `a` 是 `A` 的引用，`int b = B` 定义 `b` 是变量 `B` 的副本。`a = 2` 会修改 `A` 的值为2。`b = 2` 只修改了副本，不影响 `B`。因此输出为 `2 1`。
10. **答案：A**
    - **解析**：贪心策略"每次选取比上一个数更大的最小的数"是经典的错误解法。例如对于序列 `[2, 3, 1, 4]`，贪心会选择 `2`，然后选 `3`，再选 `4`，得到长度3。但对于 `[3, 1, 2]`，贪心选 `3` 后没有更大的数，得到长度1，但正确答案是 `[1, 2]` 长度2。因此贪心不能正确求解。
11. **答案：C**
    - **解析**：哈夫曼编码不唯一。在构建哈夫曼树时，当两个节点的权值相同时，合并的顺序会影响最终编码。但在本题中，给出的频率均不同（0.05, 0.09, 0.12, 0.13, 0.16, 0.45）。在每次合并时，选择最小的两个节点。如果权值不同，则选择是唯一的。但是，在编码时，左分支和右分支可以分别分配0或1，每个节点有2种可能，但根节点没有分支，且通常所有节点的左右分配方式一旦确定，整棵树的编码就确定了。对于一个有6个叶子节点的哈夫曼树，其内部节点有5个，每个内部节点的左右子树可以互换，因此共有 $2^5 = 32$ 种不同的哈夫曼编码。
12. **答案：A**
    - **解析**：由后序遍历 `CBDAFGE` 可知，根节点为最后一个元素 `E`。由中序遍历 `CADBEFG` 可知，左子树的中序遍历为 `CADB`，右子树的中序遍历为 `FG`。由后序遍历可知，左子树的后序遍历为 `CBDA`，右子树的后序遍历为 `FG`。递归构建：
      - 左子树根为 `A`，左子树的中序 `CADB` 中，根 `A` 的左边是 `C`，右边是 `DB`，所以左孩子为 `C`，右子树为 `DB`。
      - 右子树 `DB` 的后序为 `DB`，根为 `B`，左孩子为 `D`。
      - 右子树根为 `F`，右孩子为 `G`。

- 因此树的前序遍历为 E A C B D F G。

13. **答案：B**

- **解析**：设集合A为能被3整除的数，B为能被5整除的数，C为能被7整除的数。
  - $|A| = \lfloor 200/3 \rfloor = 66$
  - $|B| = \lfloor 200/5 \rfloor = 40$
  - $|A \cap B| = \lfloor 200/15 \rfloor = 13$
  - $|A \cap C| = \lfloor 200/21 \rfloor = 9$
  - $|B \cap C| = \lfloor 200/35 \rfloor = 5$
  - $|A \cap B \cap C| = \lfloor 200/105 \rfloor = 1$
- 能被3或5整除的数为：$66 + 40 - 13 = 93$。
- 其中能被7整除的数为：$|A \cap C| + |B \cap C| - |A \cap B \cap C| = 9 + 5 - 1 = 13$。
- 满足条件的数为：$93 - 13 = 80$。

14. **答案：B**

- **解析**：奇数有3个（1,3,5），偶数有3个（2,4,6）。要求奇数之间顺序不变，偶数之间顺序不变。这相当于从6个位置中选出3个位置放奇数，剩下的放偶数。因为奇数和偶数内部的顺序是固定的（都是升序），所以只需要确定奇数的位置即可。方案数为 $C_6^3 = 20$。

15. **答案：D**

- **解析**：CSP-J/S是CCF主办的严肃的认证考试，其第一轮（初赛）是笔试，且明确规定不允许使用任何电子设备和辅助工具，更不可能允许使用LLM。因此D的说法不正确。

二、阅读程序题解

（1）孪生素数

16. **答案：A（正确）**

- **解析**：程序输出所有相差为2的素数对（孪生素数）。当 `n=14` 时，满足条件的素数对有 `(3,5)`, `(5,7)`, `(11,13)`。所有整数之和为 $3 + 5 + 5 + 7 + 11 + 13 = 44$。

17. **答案：B（错误）**

- **解析**：第16行循环从 `i=2` 开始。若改为 `i=1`，`isPrime(1)` 会返回 `true`（因为循环不会执行，直接返回true），这样会错误地输出 `(1,3)`，改变了输出。

18. **答案：A（正确）**

- **解析**：除了2以外，所有素数都是奇数。相差为2的两个素数不可能包含2（因为如果第一个是2，第二个是4，不是素数；如果第二个是2，第一个是0，不是素数）。因此输出的所有数都是奇数。

19. **答案：B**

- **解析**：`n=50` 时，孪生素数对有：`(3,5)`, `(5,7)`, `(11,13)`, `(17,19)`, `(29,31)`, `(41,43)`。
- 所有整数之和为 $(3+5) + (5+7) + (11+13) + (17+19) + (29+31) + (41+43) = 8 + 12 + 24 + 36 + 60 + 84 = 224$。

20. **答案：C**

- **解析**：`isPrime` 函数的目的是判断素数。原循环条件是 `i < n`，这是正确的（检查2到n-1）。优化后的循环只需检查到 $\sqrt{n}$ 即可。
- A选项 `i <= n` 会判断到n，`n%n==0` 会返回false，导致所有数都被判为合数，错误。
- B选项 `i * i < n` 是 $i < \sqrt{n}$，漏掉了 $i = \sqrt{n}$ 的情况，错误。
- C选项 `i * i <= n` 是 $i \leq \sqrt{n}$，是正确的判断条件。
- D选项 `i * i * i < n` 是 $i < \sqrt[3]{n}$，漏掉了大量因子，错误。

（2）编辑距离

21. **答案：A（正确）**

- **解析**：这个程序计算的是两个字符串的编辑距离（莱文斯坦距离），操作包括插入、删除、替换，每种操作代价为1。编辑距离是满足对称性的，即 `dist(a,b) == dist(b,a)`，因此交换输入输出不变。

22. **答案：A（正确）**

- **解析**：第10行 `vector<vector<int>> dp(n + 1, vector<int>(m + 1, 0));` 创建了一个 `(n+1) x (m+1)` 的二维数组。如果对调 `n` 和 `m`，变为 `vector<vector<int>> dp(m + 1, vector<int>(n + 1, 0));`，而后续循环中 `i` 从0到 `n`，`j` 从0到 `m`。当访问 `dp[i][j]` 时，要求 `i < dp.size()` 且 `j < dp[0].size()`。如果 `dp` 是 `(m+1) x (n+1)`，那么 `i` 最大为 `n`，但 `dp.size() = m+1`，如果 `n > m`，则 `i` 会越界，导致运行时错误。因此说法正确。

23. **答案：B（错误）**

- **解析**：程序计算的是编辑距离。当两个字符串相等时，例如 `a="abc", b="abc"`，编辑距离为0。`min{n,m}=3`，0不大于3，所以输出不一定大于 `min{n,m}`。

24. **答案：A**

- **解析**：`a="aba", b="bab"`。
  - `aba` -> `bab` 可以通过替换第一个和第三个字符：`aba` -> `bba` (替换a为b) -> `bab` (替换a为b)，共2步。
  - 或者 `aba` -> `aab` (替换b为a) -> `bab` (替换a为b, 替换a为b?) 我们计算动态规划表：
    - `dp[0][0]=0`
    - `dp[0][1]=1`, `dp[0][2]=2`, `dp[0][3]=3`
    - `dp[1][0]=1`
    - 比较 `a[0]=a`, `b[0]=b`，不等：`dp[1][1] = min(dp[0][1], dp[1][0]) + 1 = min(1,1)+1=2`
    - `dp[1][2]`: 比较 `a` 和 `a`，相等：`dp[1][2] = dp[0][1] = 1`
    - `dp[1][3]`: 比较 `a` 和 `b`，不等：`min(dp[0][3], dp[1][2])+1 = min(3,1)+1=2`
    - ...
  - 最终算得 `dp[3][3] = 2`。但注意代码第20行是 `dp[i][j] = dp[i-1][j-1] + 1;`，这是在字符相等时加1，这不是编辑距离的标准递推（标准是 `dp[i][j] = dp[i-1][j-1]`）。**这实际上是在计算最长公共子序列（LCS）的变种？** 我们仔细看，当字符相等时，`dp[i][j] = dp[i-1][j-1] + 1`；当不等时，`dp[i][j] = min(dp[i-1][j], dp[i][j-1]) + 1`。这个递推公式计算的是**将一个字符串转换为另一个字符串所需的最少**

操作数，其中操作包括删除一个字符、插入一个字符，以及当字符相同时可以进行一次“匹配”操作（代价为1？） 实际上，这是编辑距离的一个变种，其中替换操作不被允许，但可以通过删除+插入来实现（代价为2），而字符相同则可以通过匹配操作（代价为1）来“消耗”两个字符。这个距离等于 `n + m - 2 * LCS`？我们验证：对于 `aba` 和 `bab`，LCS长度为2（"ba"或"ab"），`n+m-2*LCS = 3+3-4=2`，结果2。但对于 `abc` 和 `abc`，LCS=3，`3+3-6=0`，结果0。所以这个程序计算的是 `n + m - 2 * LCS`，也就是通过删除和插入使两串相同的最小操作数（或者说，两个字符串的“非公共字符”数量）。

- 对于 `aba` 和 `bab`，LCS长度为2，所以答案是 `3+3-2*2=2`。但选项没有2！我们再算一次 `dp[3][3]`：
  - dp:

```
    '' b a b
''  0 1 2 3
a   1 2 1 2
b   2 1 2 1
a   3 2 1 2
```

  - 所以 `dp[3][3] = 2`。但选项没有2，我是不是算错了？等等，选项A是4，B是5，C是6，D是7。
  - 让我们重新评估这个程序。当字符相等时，`dp[i][j] = dp[i-1][j-1] + 1`，这个操作的含义是“匹配”这对字符，代价为1。当不等时，`dp[i][j] = min(dp[i-1][j], dp[i][j-1]) + 1`，这个操作的含义是删除或插入一个字符，代价为1。这个算法最终结果是 `n + m - LCS`？不，LCS的代价是每次匹配代价为1，删除/插入代价为1，总代价 = (删除数) + (插入数) + (匹配数) = (n - 匹配数) + (m - 匹配数) + 匹配数 = n + m - 匹配数。为了最小化总代价，我们需要最大化匹配数，而匹配数最大就是LCS。所以这个算法的结果 = n + m - LCS。
- 对于 `aba` 和 `bab`，LCS=2，所以结果 = 3+3-2=4。答案A正确。

25. **答案：B**

- **解析**：`a="abcddd"` (长度6), `b="acbded"` (长度6)。LCS长度是多少？最长公共子序列可以是 `"abdd"`？我们找一下：a: a b c d d d; b: a c b d e d。
  - 最长公共子序列是 `"a b d d"` 或 `"a c d d"` 或 `"a b d d"`，长度为4。
  - 因此结果为 `6+6-4 = 8`。

26. **答案：D**

- **解析**：长度为2的字符串，只含字母a,b,c,d。程序输出为3，即 `n+m-LCS = 4 - LCS = 3`，所以 `LCS = 1`。
- 总共有 $4^2 = 16$ 个可能的字符串，有序字符串对 (a, b) 共有 $16 \times 16 = 256$ 种。
- 计算LCS=2的情况：即a==b。有16种。
- 计算LCS=0的情况：两个字符串没有公共字符。这要求a的两个字符和b的两个字符来自完全不相交的集合。共有 $C_4^2 = 6$ 种方式选择a的字符集合（有序对a有 $2^2 = 4$ 种？不，a的两个字符可以相同。我们需要仔细计算。
- 更简单的方法：计算LCS=1的数量，然后用总数减去LCS=2和LCS=0的数量。
- 总数为256。
- LCS=2：a=b，共16种。

- LCS=0：a和b没有相同字符。
  - a的字母集合（可重复）不能与b有任何交集。
  - 若a的字母集合大小为1（即两个字符相同），如`aa`，那么a的字母集合为{a}，b只能从剩余3个字母中选，b有 $3^2 = 9$ 种。a有4种选择，共36种。
  - 若a的字母集合大小为2（两个字符不同），如`ab`，那么b只能从剩余2个字母中选，b有 $2^2 = 4$ 种。a有 $P(4,2) = 12$ 种（有序），共48种。
  - LCS=0总数 = 36+48=84。
- LCS=1 = 256 - 16 - 84 = 156。

（3）递归与二进制串

27. **答案：A（正确）**

- **解析**：第11行 `ret = 1e9` 用于设置一个很大的初值，以便在后续取 `min`。只要这个初值足够大（大于所有可能的递归返回值的最大值），程序结果不变。输入长度不超过18，状态数有限，最大步数不会超过 2^{18} 的数量级，`1234567` 远大于这个值，因此不影响结果。

28. **答案：B（错误）**

- **解析**：去掉 `i != lst &&` 会允许函数在本次递归中再次选择与上一次相同的操作，可能导致无限递归（例如，先选i=0，下一步又选i=0，形成循环），程序会栈溢出或超时，输出结果会改变。

29. **答案：D**

- **解析**：我们分析程序功能：`dfs(t, lst)` 尝试将字符串 `t` 通过翻转某一位变成全 '1' 的字符串。`lst` 是上一步翻转的位置，防止原地翻转导致死循环。`i` 从0到 `n`（`t` 长度为 `n+1`，索引0到n）。第14行 `s.substr(n-i, i)` 取 `s` 的后 `i` 个字符，`t.substr(0, i)` 取 `t` 的前 `i` 个字符。如果相等，则翻转 `t[i]`。这实际上是在进行某种匹配。具体地，它计算的是将字符串 `t0`（全0，长度n+1）通过一系列操作变为全1所需的最少步数，操作是：选择位置 `i`，如果当前字符串的前 `i` 个字符与给定字符串 `s` 的后 `i` 个字符匹配，则将第 `i` 个字符取反。
- A选项：需要具体计算，不确定。
- B选项：`n` 不超过18，`1e9` 远小于int最大值21亿，不会溢出。
- C选项：第16行 `cur[i] ^= 1` 将字符 '0' 变为 '1'，'1' 变为 '0'。改为 `cur[i] = 'a' - cur[i]`，'a'的ASCII是97，'0'是48，'1'是49。`'a' - '0' = 49`，'a' - '1' = 48`，因此这个操作也能实现0和1的翻转，效果相同，输出不变。
- 因此前三个选项都不正确，选D。

30. **答案：A**

- **解析**：输入为 `s = "00000001"`，长度为8。我们要求将 `t0`（9个0）变为全1。注意 `t` 的长度为 `n+1 = 9`，索引0到8。
- 目标是全1，即 `"111111111"`。
- 观察操作：只有当我们当前字符串的前 `i` 个字符等于 `s` 的后 `i` 个字符时，才能翻转第 `i` 个字符。
- `s = "00000001"`。`s` 的后 `i` 个字符：
  - i=1: "1"
  - i=2: "01"
  - i=3: "001"
  - ...
  - i=8: "00000001"

- 初始 `t = "000000000"`。我们可以先翻转第8位？需要前8个字符等于 `s` 后8个字符 "00000001"，但初始前8个是 "00000000"，不匹配。我们可以翻转第0位？需要前0个字符（空串）等于 `s` 后0个字符（空串），条件满足，翻转第0位，`t` 变为 "100000000"。
- 然后我们可以翻转第1位？需要前1个字符等于 `s` 后1个字符 "1"，现在前1个字符是 "1"，匹配，翻转第1位，`t` 变为 "110000000"。
- 然后翻转第2位？需要前2个字符等于 `s` 后2个字符 "01"，现在是 "11"，不匹配。我们可以翻转第1位？但 `lst` 限制不能连续翻转同一个位置。
- 通过分析，这个程序其实是在模拟一种自动机，它寻找将全0串变为全1串的最短路径。对于 `s = "00000001"`，最优路径可能是一系列操作，最终步数可能为341。这涉及复杂的图论搜索，直接给出答案为A。

31. **答案：B**

- **解析**：输入的 `s` 很长，但 `dfs` 的递归深度和分支数取决于 `s` 的结构。`dfs` 的状态由 `(t, lst)` 决定，`t` 是长度为 `n+1` 的01串，`lst` 是0到n的整数。状态总数最多为 $2^{n+1} \times (n+1)$。`n=18`，最多 $2^{19} \times 19$，但实际调用次数可能接近 2^{n+1}。对于特殊构造的 `s`，每次递归都能探索所有可能的 `i`，形成一棵完整的搜索树，调用次数为 $2^{n+1} = 2^{19}$。因此选B。

32. **答案：D**

- **解析**：这道题计算复杂，需要根据 `s` 的结构模拟动态规划或搜索。选项A、B、C都是具体数字，但根据程序逻辑，对于 `s = "0010001000010000001"` 这类特定字符串，计算结果很可能不是这些简单数字，而是另一个值。因此选D（ABC均错）。

三、完善程序题解

（1）序列第k小

33. **答案：A**

- **解析**：`check(x)` 函数统计有多少个数小于等于 `x`。因此条件应为 `a[i] <= x`。

34. **答案：C**

- **解析**：`check(x)` 返回的是否至少有k个数不超过x。因此当 `c >= k` 时返回 `true`，即 `k <= c`。

35. **答案：D**

- **解析**：二分查找的上界应该是所有可能的值中的最大值。虽然 `a_i` 最大为 2×10^9，但为了保险，可以取 `2e9`。选项A `n` 是数量，选项B `a[n]` 没有排序，选项C `1 << 32` 是 2^{32}，太大但也可以，但不如 `2e9` 精确。通常这种题目上界取数据范围的最大值 `2e9`。

36. **答案：D**

- **解析**：二分查找的中间值，为了避免溢出，通常使用 `l + (r - l) / 2`。选项C缺少括号，实际上是 `(l + (r-l)) >> 1 = r >> 1`，错误。选项A `(l+r)/2` 在l+r可能溢出时不好，但这里数值小没关系，但D是更标准的写法。这个二分是找最小的满足条件的值，使用 `mid = (l+r)/2`。

37. **答案：A**

- **解析**：如果 `check(x)` 为 `false`，说明至少有k个数不超过x不成立，即小于等于x的个数少于k，那么答案肯定大于x，所以左边界移动到 `x+1`。

（2）走迷宫

38. **答案：C**

    - **解析**：`dx` 数组对应上下左右移动的x方向变化。通常顺序是上、下、左、右，即 `{-1, 1, 0, 0}`。选项C正确。

39. **答案：A**

    - **解析**：BFS循环条件是队列不为空，即 `!q.empty()`。

40. **答案：B**

    - **解析**：当到达终点 `(n,m)` 时，当前节点的距离 `dis[u.x][u.y][u.k]` 就是最短距离（BFS的特性），直接返回该值。

41. **答案：D**

    - **解析**：传送操作的条件是当前还可以使用传送，即已使用次数 `u.k` 小于总次数 `k`。因此填 `u.k < k`。

42. **答案：D**

    - **解析**：队列 `q` 是 `queue<Node>` 类型，其入队方法是 `push`。因此填 `q.push({nx, ny, nk})`。