SCP-S 2022 模拟赛

一、 单项选择题（共 15 题，每题 2 分，共计 30 分；每题有且仅有一个正确选项）

1. $(14122)_8 =$

   A. $(44544)_6$
   B. $(110001010010)_2$
   C. $(6326)_{10}$
   D. $(8477)_9$

> 【答案】D

2. 一张未经压缩的1024×512像素的 BMP 图像，压缩为 JPG 后图像大小是 64KB，压缩比为$8:1$，该图像的颜色至多

   A. 8 色
   B. 256 色
   C. 16 位色
   D. 24 位色

> 【答案】B
>
> 【解析】设位深度为 x，则$\frac{1024\times256\times x}{64\times8\times1024} = \frac{1}{8}$，解得 $x = 8$，故为 8 位色，$2^8 = 256$ 色

3. Linux 中查看当前路径使用的命令是

   A. pwd
   B. ps
   C. mv
   D. fin

> 【答案】A

4. 以下关于C++ 语法说法正确的是

   A. template后可接函数
   B. list 只能沿一个方向遍历
   C. 交换两个 unordered_multimap不可以使用 swap()
   D. 一个名为 s 的 set 执行一次 lower_bound(s.begin(),s.end(),value) 的时间复杂度是$O(\log n)$的

> 【答案】A
>
> 【解析】list 可沿2个方向遍历，故B错误；stl中的容器都自带swap函数，故C错误；set查询正确写法应为s.lower_bound(value)，上述写法是O(n)的，故D错误。

5. 下列关于计算机的说法正确的是

   A. 迄今为止没有一个中国国籍的人获得过图灵奖
   B. 计算机最早运用于过程控制
   C. 图灵奖的建立标志着信息论研究的开端
   D. 冯·诺依曼提出的计算机硬件设备中有控制器

> 【答案】D

【解析】姚期智是至今唯一一个中国国籍获得图灵奖的人，故A错误；计算机最早运用于数值计算（和军事研究），故B错误；人们通常将香农于1948年10月发表于《贝尔系统技术学报》上的论文《通信的数学理论》作为现代信息论研究的开端；故 C错误；冯·诺依曼提出的计算机硬件设备由存储器、运算器、控制器、输入设备、输出设备五部分构成，故D正确。

6. 以下属于解释型语言的是

A. Java
B. C++
C. 汇编语言
D. VB

【答案】A

【解析】C++、VB属于编译型语言，汇编属于低级语言。

7. 有一个算法，满足$T(n)=3T(\frac{n}{3})+n^2$，其时间复杂度为

A. $O(n^2\log_3 n)$
B. $O(n^2\log_2 n)$
C. $O(n^2)$
D. 以上都不对

【答案】C

【解析】主定理易证

8. 有一颗二叉树，中序遍历为DBGEHACF，后序遍历为DGHEBFCA，则前序遍历为

A. ABCDEFGH
B. ABDEGHCF
C. ABDEFGHC
D. ABDEGHFC

【答案】B

9. 现任 ccf 秘书长是

A. 杜子德
B. 图灵
C. 王选
D. 唐卫清

【答案】D

10. 你正在打怪兽，每次攻击在$[5,15]$中随机，怪物的防御在$[0,10]$中随机，当且仅当攻击大于防御时会造成攻击-防御的伤害，一次伤害的期望。

A. $\frac{120}{23}$
B. $\frac{175}{8}$
C. $\frac{125}{24}$
D. $\frac{175}{24}$

【答案】C

【解析】设a为当前造成的伤害，分$[5,10]$,$(10,15]$两部分计算，显然两者概率相等。

$a\in[5,10]$时，有$\frac{a}{10}$的概率造成伤害，造成伤害的期望为$\frac{a}{2}$。所以这部分期望$\frac{\int_5^{10}\frac{a}{10}\times\frac{a}{2}}{5}=\frac{35}{12}$

> $a \in (10, 15]$时，伤害为$期望攻击 - 期望防御 = \frac{25}{2} - 5 = \frac{15}{2}$
>
> 故伤害期望为$\frac{125}{24}$

11. 以下哪种排序算法是稳定的

   A. 快速排序

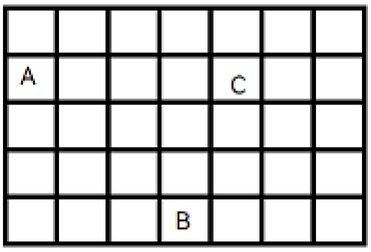
   B. 希尔排序

   C. 堆排序

   D. 计数排序

> 【答案】D
>
> 【解析】稳定性是指相等的元素经过排序之后相对顺序是否发生了改变。上述排序中只有计数排序是稳定的。

12. 一只马（众所周知，马走日）从以下哪个点开始，可以跳遍下图这半张棋盘，每个点只经过一次，并且最后一步跳回起点？



   A. A点

   B. B点

   C. C点

   D. 以上都不行

> 【答案】D
>
> 【解析】可以数出图上一共有35个格点，对所有的点进行黑白染色，相邻的点颜色不同，若点A为白色，则B、C都为白色，共有17个白点，18个黑点，由于马走日字型，走的颜色黑白交替，则不可能从一个白点出发走完整张图

13. 一个n个点的笛卡尔树每个点的平均深度期望为

   A. $2\sum_{i=1}^{n}\frac{1}{i}+1$

   B. $2\sum_{i=1}^{n}\frac{n-i+1}{i}-1$

   C. $\frac{2}{n}\sum_{i=1}^{n}\frac{1}{i}+1$

   D. $\frac{2}{n}\sum_{i=1}^{n}\frac{n-i+1}{i}-1$

> 【答案】D
>
> 【解析】根据期望线性，其平均深度就是每个点深度期望求和最后乘上$\frac{1}{n}$。
>
> 而$E(dep_u) = \sum_v v是u祖先的概率$
>
> 考虑什么情况下j是i的父亲
>
> 其实就是j为[j,i]的区间最值，概率为$\frac{1}{|j-i+1|}$
>
> 所以

$$
\begin{aligned}
ans &= \frac{\sum_{i=1}^{n}(\sum_{j=1}^{i}\frac{1}{j}) + (\sum_{j=1}^{n-i+1}\frac{1}{j}) - 1}{n} \\
&= 2\sum_{i=1}^{n}\sum_{j=1}^{i}\frac{1}{j} \\
&= \frac{2}{n}\sum_{i=1}^{n}\frac{n-i+1}{i} - 1
\end{aligned}
$$

14. 三个点组成一个三角形 ABC，有一个小猴子开始在 A 点上，每次随机往与当前点相邻的一个点跳过去，求期望多少次跳到 B 点。

    A. 1

    B. 1.5

    C. 2

    D. 3

> 【答案】C
>
> 【解析】设需要跳 x 次，容易列出 $x = \frac{1}{2} \times 1 + \frac{1}{2} \times (1 + x)$，解得 $x = 2$

15. 2021 年的 IOI 是第几届

    A. 30

    B. 31

    C. 32

    D. 33

> 【答案】D

二、 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√， 错误填×；除特殊说明外，判断题 2 分，选择题 3 分，共计 40 分）

（1）阅读以下程序，完成第 16~21 题。

```
1 #include <bits/stdc++.h>
2 using namespace std;
3 int main() {
4     int n; cin>>n;
5     double x=1;
6     while (fabs(x*x-n)>=0.5) x=(x+1.0*n/x)/2;
7     printf("%d\n", (int)x);
8     return 0;
9 }
```

保证读入的n为int范围内的正整数，无前导0

· **判断题**

16. （1.5分）当输入为20时输出为 5
17. （1.5分）输出的答案不可能为 0
18. 若第 5 行改为 x=n 则还会输出和原来一样的结果
19. 输出的答案有可能超过 n

· **选择题**

20. 若输入的数的长度为 len，则输出的长度为

- **A.** len
- **B.** $\lfloor \frac{len}{2} \rfloor$
- **C.** $\lceil \frac{len}{2} \rceil$
- **D.** $\lfloor \sqrt{len} \rfloor$

21. 当输入为 `123` 时输出为多少

- **A.** 10
- **B.** 11
- **C.** 12
- **D.** 13

> 【答案】×√√×CA
>
> 【解析】
>
> 容易结束时 $x \times x - n \le 0.5$发现题目让我们求平方根下取整
>
> 1) $\lfloor \sqrt{20} \rfloor = 5$
>
> 2) 正整数平方根下取整还为正整数
>
> 3) x一开始等于1或n运算一次后x的值相同，并且若n>1第一次做完不会退出，n=1时x=1和x=n相同
>
> 4) 正整数平方根不可能超过自己
>
> 5)平方根的长度为除以2上取整
>
> 6)计算平方根下取整，把选项带进去即可

(2) 阅读以下程序，完成第 22~27 题。

```cpp
#include <bits/stdc++.h>
using namespace std;
inline int read() {
    int x = 0, f = 1; char c = getchar();
    while (c < '0' || c > '9') {if (c == '-') f = -f; c = getchar();}
    while (c >= '0' && c <= '9') {x = (x << 3) + (x << 1) + (c ^ 48); c = getchar();}
    return x * f;
}
typedef long long LL;
const int N = 2e5 + 5;
int n, m, k, ans;
int a[N];
multiset <int> S;
inline void Solve() {
    n = read(); m = read(); k = read();
    for (int i = 1; i <= n; i++) S.insert(read());
    for (int i = 30; ~i; i--) {
        if (S.size() < k) break;
        int x = 1 << i; LL cnt = 0;
        auto it = S.lower_bound(x), tmp = it;
        vector <int> Now;
        for (; it != S.end(); it++) Now.push_back(*it);
        if (Now.size() < k) {
            it = tmp;
```

```
25                  while (Now.size() < k) {
26                      --it;
27                      Now.push_back(*it);
28                  }
29              }
30              for (int p : Now) if (p < x) cnt += x - p;
31              if (m < cnt) {
32                  for (int p : Now) {
33                      if (p < x) continue;
34                      S.erase(S.find(p));
35                      S.insert(p & (x - 1));
36                  }
37                  continue;
38              }
39              ans |= x; m -= cnt;
40              while (S.size()) S.erase(S.begin());
41              for (int p : Now) {
42                  int u;
43                  if (p < x) u = 0;
44                  else u = (x - 1) & p;
45                  S.insert(u);
46              }
47          }
48          printf("%d\n", ans);
49      }
50      int main() {
51          int _ = 1;
52          while (_--) Solve();
53          return 0;
54      }
```

保证读入的数都是非负整数且都小于等于 2^{30}。

- **判断题**

22. （1.5分）如果读入的 n超过$2 \times 10^5 + 5$，会出现数组越界的情况
23. 第 20 行中S.lower_bound(x) 改为 S.find(x) 后程序不会出错
24. 第 22 行中it != S.end() 改为it != (++S.rbegin()) 后程序不会出错
25. 第 40 行中while (S.size()) S.erase(S.begin()) 改为S.clear() 后程序不会出错

- **选择题**

26. 若读入数据如下，则输出的结果为

```
1  4 8 2
2  1 2 4 8
```

A. 10
B. 15
C. 11
D. 16

27. 若读入数据如下，则输出的结果为

```
31 4 2
1 2 4 8 …… 536870912 1073741824
```

A. 4
B. 2147483647
C. 1073741824
D. 8

> 【答案】×××√AD
>
> 【解析】
>
> 题目来源:Link。
>
> 1.根本没用到数组。~~直到我搬题的时候才发现 a 数组没有被使用~~。
>
> 2.改为 `s.find(x)` 后如果没有找到会返回 `s.end()`。
>
> 3.`s.rbegin()` 的类型为 `reverse_iterator<_Rb_tree_const_iterator<int> >`，不能直接与作为 `_Rb_tree_const_iterator<int>` 的 `s.end()` 作比较，否则会出现编译错误，另外，`s.rbegin()` 相当于翻转了整个 `multiset`，所以当 `s.rbegin()++` 时返回的不是 `s.end()`。
>
> 4.第 40 行起到的是清空 `multiset` 的作用。
>
> 5.手动模拟即可。
>
> 6.发现 m 不够大所以在 i 枚举到 3 之前都会进入第 31 行的 `if`，当 i 为 3 的时候 `multiset` 中不为 0 的元素只有 `1 2 4 8`。发现此时 m 足够大，答案或上 8。之后 m 变为 0，不再操作。

（3）阅读以下程序，完成第 28~32 题。

```cpp
#include <bits/stdc++.h>
#define int long long

using namespace std;

int read() {
    int x=0, f=0; char c=getchar();
    while (!isdigit(c)) f|=c=='-', c=getchar();
    while (isdigit(c)) x=(x<<3)+(x<<1)+(c^48), c=getchar();
    return f ? -x : x;
}

const int N=1e5+10;
int n, l, r, len1, len2, x[N], y[N];
vector<int> g1[N], g2[N];

int query1(int x, int y1, int y2) {
    if (x>len1) return 0;
    int ans1=0, ans2=0;
    ans1=upper_bound(g1[x].begin(), g1[x].end(), y2)-g1[x].begin();
    ans2=lower_bound(g1[x].begin(), g1[x].end(), y1)-g1[x].begin();
    return ans1 - ans2;
}

int query2(int x, int y1, int y2) {
```

```cpp
26      if (x>len2 || !g2[x].size()) return 0;
27      int ans1=0, ans2=0;
28      ans1=upper_bound(g2[x].begin(), g2[x].end(), y2)-g2[x].begin();
29      ans2=lower_bound(g2[x].begin(), g2[x].end(), y1)-g2[x].begin();
30      return ans1 - ans2;
31  }
32
33  int sum(int s) {
34      if (s==0) return 0;
35      int ans=0;
36      for (int i=1; i<=n; i++)
37          for (int j=1; j*j<=s; j++) {
38              int len=s/j;
39              ans+=query1(x[i]+j, y[i]+1, y[i]+len);
40              ans+=query2(y[i]+j, x[i]+sqrt(s)+1, x[i]+len);
41          }
42      return ans;
43  }
44
45  signed main() {
46      n=read(), l=read(), r=read();
47      for (int i=1; i<=n; i++) {
48          x[i]=read(), y[i]=read();
49          g1[x[i]].push_back(y[i]);
50          g2[y[i]].push_back(x[i]);
51          len1=max(len1, x[i]);
52          len2=max(len2, y[i]);
53      }
54      for (int i=1; i<=len1; i++) sort(g1[i].begin(), g1[i].end());
55      for (int i=1; i<=len2; i++) sort(g2[i].begin(), g2[i].end());
56      printf("%lld\n", sum(r)-sum(l-1));
57      return 0;
58  }
```

保证输入的所有数为≤ 10000的正整数

· **判断题**

28. （1.5分）若删除第34行程序依然正确
29. 若删除第18行的!g1[x].size()条件程序依然正确
30. 该代码的时间复杂度为$O(n\sqrt{l}\log n)$
31. sum 函数的返回值可能为$\frac{n(n+1)}{2}$

· **选择题**

32. 当输入如下时，输出为

```
4 1 100
1 1
1 2
2 1
2 2
```

A. 0
B. 1
C. 2
D. 4

33. 当输入第一行为 `n 3 7`（$n \geq 3$），且第 $i \in [2, n+1]$ 行输入为 `i i`，则输出应为

A. n
B. $n-2$
C. $2n$
D. $2n-2$

> 【答案】√√××BB
>
> 【解析】
>
> 原题：AT1004
>
> 我们首先要搞懂题目在让我们求什么，容易发现读入了 n 个坐标 x, y。
>
> 发现输出是 $sum(r) - sum(l-1)$，容易想到前缀和，就是小于等于 r 的答案减去小于 l 的答案，说明题目让我们求的是在 $[l, r]$之间的答案
>
> 先看 query1，发现是在查询很坐标为 x 时，纵坐标 $> y2$ 的个数减去 $\geq y1$ 的个数，即纵坐标在 $[y1, y2]$ 之间的点的个数，query2同理。
>
> 理解 query 后容易发现 sum(x) 求的是两个点的围成的面积 $<= x$ 的点对数，并且保证一个点在另一个点的右上方
>
> 所以题目让我们求得时两点构成的面积在 $[l, r]$ 之间的答案，并且一个点在另一个点的右上方，然后题目做起来就很简单了
>
> 1. 当 $s = 0$ 时，第二重循环不会被做到，相当于直接返回 $ans = 0$
> 2. 若 vector 为空，则两次查询都会返回 $g1.end()$，相减任为 0
> 3. 复杂度应为 $O(n\sqrt{r}\log n)$
> 4. 查询的是点对的个数，最多 $\frac{n(n-1)}{2}$ 对点对
> 5. 直接带入即可（知道代码求什么的话一眼就可看出答案）
> 6. 构成的面积要在 $[3, 7]$ 之间，特殊的点坐标使两点之间面积为 $(i-j)^2$，应为坐标为正数，所以满足条件的点对只有 $[i, i+2]$，共 $n-2$ 对。

三、完善程序（每小题 3 分，共计 30 分）

（1）你要前往bobo之乡。前往bobo之乡的路上有n段路，每段路的路况由三个数$s_i, k_i, {v'}_i$表示，分别为长度、风阻系数和风速(${v'}_i < 0$表示逆风)。

你以v的速度通过第i段路要花费$E = s_i \times k_i \times (v - {v'}_i)^2$的能量。

你初始有Eu的能量，问到达bobo之乡最快要多久?

提示：先假设每段路用 0 能量，每次找到性价比最高的消耗能量，容易发现最后各路段性价比会相同。而性价比即为 $E(v)/t(v)$ 的导数。因其单调递减，二分查找这个导数即可。

```
#include <cstdio>
#include <cstring>
#include <cmath>
#include <algorithm>
#include <iostream>
#define enter putchar('\n')
```

```
#define space putchar(' ')
using namespace std;
typedef long long ll;
template <class T>
void read(T &x){
    char c;
    bool op = 0;
    while(c = getchar(), c < '0' || c > '9')
        if(c == '-') op = 1;
    x = c - '0';
    while(c = getchar(), c >= '0' && c <= '9')
        x = x * 10 + c - '0';
    if(op == 1) x = -x;
}
template <class T>
void write(T x){
    if(x < 0) putchar('-'), x = -x;
    if(x >= 10) write(x / 10);
    putchar('0' + x % 10);
}

const int N = 10005, INF = 0x3f3f3f3f;
int n;
double E, s[N], k[N], u[N];

double getv(double x, int i){
    double l = max(u[i], double(0)), r = 100005, mid;
    int cnt = 60;
    while(cnt--){
        mid = (l + r) / 2;
        if(__________(34)__________ > -s[i]) l = mid;
        else r = mid;
    }
    mid = (l + r) / 2;
    return (l + r) / 2;
}
double calc(double x){
    double sum = 0;
    for(int i = 1; i <= n; i++){
        double v = getv(x, i);
        sum += __________(35)__________;
    }
    return sum;
}

int main(){

    scanf("%d%lf", &n, &E);
    for(int i = 1; i <= n; i++)
        scanf("%lf%lf%lf", &s[i], &k[i], &u[i]), k[i] *= __________(36)__________;
    double l = __________(37)__________, r = 0, mid;
    int cnt = 100;
    while(cnt--){
        mid = (l + r) / 2;
        if(calc(mid) <= E) l = mid;
```

```
            else r = mid;
        }
        mid = (l + r) / 2;
        double ans = 0;
        for(int i = 1; i <= n; i++)
            ans += __________(38)__________;
        printf("%.10lf\n", ans);

        return 0;
    }
```

34. **A.** `2 * k[i] * x * mid * mid * (mid - u[i])`

    B. `k[i] * x * mid * mid * (mid - u[i])`

    C. `2 * k[i] * x * mid * (mid - u[i])`

    D. `k[i] * x * mid * (mid - u[i])`

35. **A.** `k[i] * (v - u[i]) * (v - u[i])`

    B. `k[i] * v * v`

    C. `2*k[i] * (v - u[i]) * (v - u[i])`

    D. `2*k[i] * v * v`

36. **A.** `i`

    B. `k[i]`

    C. `s[i]`

    D. `u[i]`

37. **A.** `INF`

    B. `-INF`

    C. `1`

    D. `-1`

38. **A.** `s[i] * k[i]`

    B. `s[i]`

    C. `s[i] * k[i] / getv(mid, i)`

    D. `s[i] / getv(mid, i)`

> 【答案】
>
> AACBD
>
> 【解析】
>
> 注意到一个性质：随着花费能量增加，性价比会越来越低。
>
> 这样的话，只要按照上面这种贪心策略，时时刻刻在性价比最高的路段花费能量（并使它的性价比降低），最后达到最优解时，**各路段性价比会一样。**
>
> 这个性价比是什么呢？如果我们对每段路画出一个t−E函数图象，表示该路段需要的时间t**与**花费的能量E**E的函数关系，那么花费一定能量e之后的“性价比”是什么呢？就是函数图像上横坐标为e处切线的斜率——导数。
>
> 那么最优解就满足——**各路段导数一样！**

> 同时，这个公共导数（是负的）绝对值越小（性价比越低），所需能量越多，总时间越小。
>
> 于是二分这个导数，求出每段速度，以此求出所需能量，和手里的总能量比较一下，就可以二分得到答案了！
>
> 以上是思路。现在开始数学。
>
> 要求出每段导数关于v的关系。
>
> 对于一段路来说
>
> dE/dt
>
> $=(dv/dt)/(dv/dE)$
>
> $=2k(v-v')$
>
> $= -2kv2(v-v')s$
>
> 然后二分公共导数x，对于每段路解方程$-2kv2(v-v')s=x$（可二分）得到v，进而求出需要的能量。

（2） yb 有n颗星星，每颗星星都有一个明亮度 A_i 。yb 时常想知道一个区间$[l,r]$内所有星星的明亮度的总和是多少。但是星星是会眨眼的，所以星星的明亮度是会变化的。有的时候，下标为$y,y+x,y+2x,y+3x,\ldots,y+kx$（$k$为最大的整数使$y+kx\le n$）的星星的明亮度会增加$z$。保证$y\le x$。

yb 是数学大神，所以请回答**她**的询问。答案要对10^9+7取模。

提示：按照x的大小进行根号分治。（代码中为了卡常所以阈值没有赋为$\sqrt{n}$）。

注意：本题中对加法取模请使用add函数。

```
#include <bitsdc++.h>
using namespace std;
inline int read() {
    int x = 0, f = 1; char c = getchar();
    while (c < '0' || c > '9') {if (c == '-') f = -f; c = getchar();}
    while (c >= '0' && c <= '9') {x = x * 10 + (c ^ 48); c = getchar();}
    return x * f;
}
inline void write(int x) {
    char Buf[11];
    int tot = 0;
    while (x) {
        Buf[++tot] = x % 10 ^ 48;
        x /= 10;
    }
    for (int i = tot; i; i--) putchar(Buf[i]);
}
const int N = 2e5 + 5, M = 455, mod = 1e9 + 7;
int n;
int a[N];
inline void add(int &X, int Y) {
    X += Y;
    if (X >= mod) X -= mod;
}
class abcdefg {
private:
    const int T = 125;
    int block;
```

```
    int bel[N], l[M], r[M], sum[M], pre[M][M], suf[M][M];
    inline int getsum(int L, int R) {
        int p = bel[L], q = bel[R], res = 0;
        if (p == q) {
            for (int i = L; i <= R; i++) add(res, a[i]);
            return res;
        }
        for (int i = L; i <= r[p]; i++) add(res, a[i]);
        for (int i = l[q]; i <= R; i++) add(res, a[i]);
        for (int i = p + 1; i < q; i++) add(res, sum[i]);
        return res;
    }
public:
    inline void init() {
        block = sqrt(n);
        for (int i = 1; i <= block; i++) {
            l[i] = r[i - 1] + 1;
            r[i] = i * block;
            for (int j = l[i]; j <= r[i]; j++) bel[j] = i;
            for (int j = l[i]; j <= r[i]; j++) add(sum[i], a[j]);
        }
        if (r[block] != n) {
            r[++block] = n;
            __________(39)__________
            for (int j = l[block]; j <= r[block]; j++) bel[j] = block;
            for (int j = l[block]; j <= r[block]; j++) add(sum[block], a[j]);
        }
    }
    inline void change(int x, int y, int z) {
        if (__________(40)__________) {
            for (int i = y; i <= n; i += x) add(sum[bel[i]], z);
            for (int i = y; i <= n; i += x) add(a[i], z);
            return;
        }
        for (__________(41)__________) add(pre[x][i], z);
        for (int i = 1; i <= y; i++) add(suf[x][i], z);
    }
    inline int query(int l, int r) {
        int res = getsum(l, r), nm = min(n, T);
        for (int x = 1; x <= nm; x++) {
            int p = (l - 1) / x + 1, q = (r - 1) / x + 1;
            if (p == q) {
                add(res, pre[x][r - (q - 1) * x]);
                add(res, __________(42)__________);
            }
            else {
                add(res, __________(43)__________);
                add(res, pre[x][r - (q - 1) * x]);
                add(res, suf[x][l - (p - 1) * x]);
            }
        }
        return res;
    }
} B;
int main() {
```

```
    n = read(); int q = read();
    for (int i = 1; i <= n; i++) a[i] = read();
    B.init();
    while (q--) {
        int op = read(), x = read(), y = read();
        if (op & 1) B.change(x, y, read());
        else {
            write(B.query(x, y));
            putchar('\n');
        }
    }
    return 0;
}
```

39. ________________

40. ________________

41. ________________

42. ________________

43. ________________

【答案】

1. `l[block] = r[block - 1] + 1`，也可以直接计算；
2. `x > T`；
3. `int i = y; i <= x; i++`；
4. `mod - pre[x][l - 1 - (p - 1) * x]`；
5. `1LL * pre[x][x] * (q - p - 1) % mod` 或将 `pre[x][x]` 改为 `suf[x][1]`。

【解析】

~~这题本来想放到分块课件里的~~。

大于 $\sqrt{n}$ 时暴力加，使用分块维护。

小于 $\sqrt{n}$ 时考虑题目性质，由于 $y \le x$，相当于每次修改的位置以 x 为周期，对 x 取模相等。

令 $b_{i,j}$ 表示周期为 i 起始点为 j 的和。

令 $pre_{i,j} = \sum_{k=1}^{j} b_{i,k}$，即前缀和，同理维护后缀和 $suf_{i,j}$。

查询的时候枚举周期大小，分两种情况，一种情况为两者处于同一周期，另一种情况为不同周期。

同一个周期可以用前缀和转化为区间的形式。

不同周期可以采用类似分块的思想，中间的周期可以直接计算，左侧用后缀和，右侧用前缀和，这题就做完了。

第一空比较明显，预处理最后一块的左端。

第二空发现下面 `query` 函数中 `pre` 的第一维永远小于等于 T。

第三空为预处理前缀和，由于加的下标为 y，应该从 $y \to x$。

第四空根据前缀和的写法不难得知，但注意不加 `mod` 会导致 `res` 变负数。

第五空上面讲了。