Практический тур №1

Задача 1 «Сокращаем перемены»

(уровень сложности: Муниципальный, 7-8 кл, элементарная)

Условие задачи:

Требуется подсчитать, на сколько минут раньше будет заканчиваться k-й урок, если все перемены сократить на m минут.

Анализ решения

Для решения задачи достаточно составить простую формулу, приняв во внимание, что перед первым уроком и после последнего перемен нет 😊
Таким образом, количество сокращаемых перемен будет k-1, а заданный урок будет заканчиваться на (k-1) * m минут раньше

Текст программы

```cpp
#include <iostream>
int main()
{
    int k, m;
    std::cin >> k >> m;
    std::cout << ((k - 1) * m) << "\n";
}
```

Задача 2 «Урок физкультуры»

(уровень сложности: Муниципальный, 7-8 кл, простая)

Условие задачи:

На уроке физкультуры тренер Андрей Сергеевич выстраивает учеников в одну шеренгу. В шеренге сначала идут мальчики, а потом девочки. При этом мальчики в шеренге стоят по убыванию роста, аналогично девочки тоже стоят по убыванию роста. Таким образом, следом за самым низким мальчиком стоит самая высокая девочка.
Андрея Сергеевича заинтересовал вопрос, какое максимальное различие в росте двух стоящих рядом учеников. Напишете программу, которая поможет Андрею Сергеевичу ответить на этот важный для него вопрос.

Анализ решения

Отсортируем школьников в соответствии с принципом, описанным в условии.
После этого осталось найти максимум разностей соседних элементов. Сделаем это линейным проходом по массиву.

Поскольку ограничения на число школьников были достаточно маленькие, можно было воспользоваться любым алгоритмом сортировки, например, сортировкой пузырьком.

В C++ имеется стандартная функция сортировки sort(), поэтому есть смысл воспользоваться этой функцией :)
Кроме этого, в данном решении демонстрируется использование вектора (std::vector) для хранения и обработки массива чисел. Еще для сортировки удобно вводимые данные объединить в пары (std::pair)

Возможны также аналогичные решения без использования векторов, но с применением стандартной функции сортировки, такой вариант предлагаем разработать самостоятельно 😊

Текст программы

```
#define TASKNAME "gym" // Для имен файлов
#include<stdio.h>
#include<algorithm>
#include<vector>

using namespace std;
typedef pair <int, int> pii;
typedef vector <pii> vpii;
```

```
int main()
{
    FILE* in;  fopen_s(&in,  TASKNAME".in",  "r");
    FILE* out; fopen_s(&out, TASKNAME".out", "w");

    int n;
    fscanf_s(in, "%d", &n);

    // vector pair <int, int> a;
    vpii a;
    for (int i = 0; i < n; i++) {
        int x, y;
        fscanf_s(in, "%d%d", &x, &y);
        a.push_back(make_pair(x, -y));
    }

    std::sort((a).begin(), (a).end());
    int ans = 0;
    for (int i = 0; i < size(a) - 1; i++)
        ans = max(ans, abs(a[i].second - a[i + 1].second));

    fprintf(out, "%d\n", ans);
    fclose(in);
    fclose(out);
    return 0;
}
```

Задача 3 «Пятница, 13-е»

(уровень сложности: Муниципальный, 7-8 кл, простая)

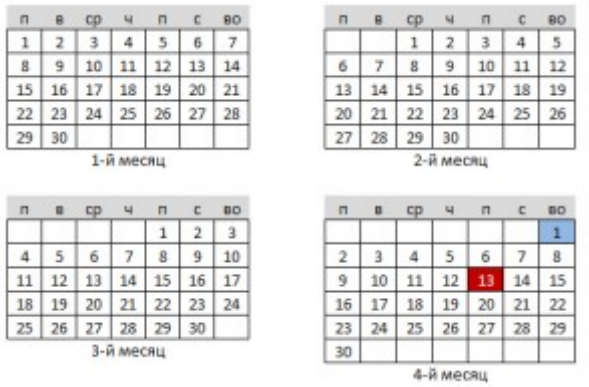
Условие задачи:

Календарь жителей планеты Плюк состоит из N месяцев, каждый месяц состоит ровно из 30 дней, неделя состоит из 7 дней. Особо несчастливыми на планете Плюк считается 13-е число месяца, если оно выпадает на пятницу.
Известно, что Новый год на планете Плюк начался в k-й по счету день недели (1-й день недели – понедельник, 2-й – вторник, 3-й – среда, … ,7-й – воскресенье).

Определите, сколько в этом году на планете Плюк будет особо несчастливых пятниц, 13-е.

Анализ решения

Рассмотрим несколько месяцев в году на планете Плюк:



Как видите:

- пятница может стоять 13-м числом, если первый день месяца — воскресенье (7-ой день недели);
- каждый последующий месяц первый день месяца сдвигается на два дня (был — понедельник, стал — среда; и т.д.)

В цикле (который перебирает все месяцы в году на планете) создадим условие, проверяющее: является ли 13-е число пятым по счету днем недели (т.е. пятницей).

Можно использовать для решения следующее условие:
```
if ((m+12) % 7 == 5) k=k+1;
```
оно будет истинно только в случае, когда m — седьмой день недели (19 % 7 == 5); k — счетчик пятниц 13-ых. m — номер дня недели для первого дня месяца: понедельник (m=1), вторник (m=2) и т.д.

Необходимо также рассмотреть еще два условия:
```
if (m == 8) m=1; if (m == 9) m=2;
```

Если в настоящий месяц первой число — это суббота или воскресенье, то после выполнения команды **m=m+2** (переход на следующий месяц) значение m может быть равным 8 или 9. Но нам нужна первая неделя месяца, чтобы проверить самое первое условие, поэтому мы переопределяем переменную m.

К сожалению, такое решение не проходит большие тесты по времени и набирает 74 балла из 100
Для решения задачи на полный балл заметим, что поскольку числа 7 и 30 — взаимно простые, то в последовательности дней недели, на которые выпадают 13-е числа месяца будет цикл длины 7: пятница, воскресенье, вторник, четверг, суббота, понедельник, среда. Поэтому для нахождения ответа достаточно найти первый месяц года, в котором будет пятница 13-е, и к ответу добавить количество оставшихся месяцев в году, деленное на 7 нацело.

Текст программы

```
#include <iostream>
int main()
{
    long long n, m, i, k;

    std::cin >> n >> k;

    k = (k + 12) % 7;
    m = 0;

    i = 1;
    while (i <= n and k != 5)
    {
        i += 1;
        k = (k + 30) % 7;
    }

    if (i > n)
        m = 0;
    else
        m = 1 + (n - i) / 7;

    std::cout << m << "\n";
}
```

Задача 4 «Сумма минимумов»

(уровень сложности: Муниципальный, 7-8 кл, сложная)

Условие задачи:

У Саши есть блокнот, состоящий из n листочков, пронумерованных от 1 до n. На i-м листочке написано целое число a_i.
Аня собирается разорвать блокнот на k частей, для этого она выбирает k-1 число $1 \le r_1 < r_2 < \ldots < r_{k-1} < n$ и разрывает блокнот так, что листки с 1 по r_1-й оказываются в первой части, листки с (r_1+1)-го по r_2-й оказываются во второй части, и т.д., последняя k-я часть содержит листки с $(r_{k-1} + 1)$-го по n-й.

После того, как Аня разорвет блокнот, Саша найдет минимальное число в каждой из получившихся частей и сложит их. Аня хочет разорвать блокнот таким образом, чтобы получившаяся сумма была как можно больше. Помогите ей выбрать способ разорвать блокнот, чтобы максимизировать сумму минимальных значений.

Анализ решения

Это наиболее сложная задача в контесте (что типично именно для последней задачи :). Для ее решения на полный балл используем динамическое программирование.

Обозначим как rmq[i][j] минимальный элемент в массиве между i-й и j-й позицией, включительно.

Для вычисления rmq[i][j] можно использовать следующие формулы:

rmq[i][i] = 1

rmq[i][j] = min(rmq[i][j – 1], a[j])

Обозначим как d[r][c] максимальную сумму минимумов, которую можно получить на отрезке с первого по r-й элемент, если разбить его на c частей.

Тогда d[r][c] = min(d[r – k][c – 1] + rmq[r – k + 1][r]) для всех k от 1 до r

Ответ содержится в элементе d[n][k]. Для восстановления самого разбиения необходимо также запоминать, для какого k достигается минимум, это позволяет восстановить ответ.

Следует также обратить внимание на возможность переполнения 32-битного типа данных в этой задаче. Те, кто использовал 32-битный тип данных, набирали 60 баллов.

В приведенном решении для формирования считывания исходных данных и формирования массива ответов используется **vector<int>**, хотя это не является обязательным условием успешного решения. Обратите также внимание на описание больших по размеру рабочих массивов данных ВНЕ функции **main()**.

Текст программы

```
#define TASKNAME "text"

#include <vector>
#include <stdio.h>
#include <algorithm>

using namespace std;

const int maxn = 400;
const int INF = 1e9;

int n, k;
long long dp[maxn][maxn];
int pr[maxn][maxn];

int main()
{
    FILE* stream;
    freopen_s(&stream, "summin.in", "r", stdin);
    freopen_s(&stream, "summin.out", "w", stdout);

    scanf_s("%d%d", &n, &k);
    vector <int> a(n);
    for (int i = 0; i < n; i++) {
        scanf_s("%d", &a[i]);
    }

    for (int i = 0; i <= k; i++)
        for (int j = 0; j <= n; j++)
            dp[i][j] = -INF;
    dp[0][0] = 0;
    for (int i = 1; i <= k; i++) {
        for (int j = 1; j <= n; j++) {
            int mn = a[j - 1];
            for (int jj = j - 1; jj >= 0; jj--) {
                if (dp[i][j] < dp[i - 1][jj] + mn) {
                    dp[i][j] = dp[i - 1][jj] + mn;
                    pr[i][j] = jj;
                }
                if (jj > 0)
                    mn = min(mn, a[jj - 1]);
            }
        }
    }
```

```
    printf("%lld\n", dp[k][n]); //%I64d
    int nowi = k;
    int nowj = n;
    vector <int> ans;
    while (nowi > 0) {
        if (pr[nowi][nowj] != 0)
            ans.push_back(pr[nowi][nowj]);
        nowj = pr[nowi][nowj];
        --nowi;
    }
    reverse((a).begin(), (a).end());
    for (int i = 0; i < size(ans); i++)
        printf("%d ", ans[i]);
    return 0;
}
```