Порядок проведення ІІ (міського/районного) етапу олімпіади з інформатики

1. **Перевірка завдань автоматична**. Учасникам олімпіади слід надсилати свої програми до автоматичної системи приймання та перевірки робіт **https://ejudge.ckipo.edu.ua** (надалі, ejudge; при цьому слід розуміти, що в Інтернеті є багато єджаджів, і потрібен саме цей, а не якийсь інший). Ця система компілює та запускає програму, що надіслав учасник олімпіади, на наборі тестів, які підготувало журі, та повідомляє учаснику результат (кількість нарахованих балів)**.** Здача розв'язків не через систему ejudge гіпотетично можлива, але лише у рідкісних особливих ситуаціях і лише з прямого дозволу журі. Врахуйте також, що при здачі в ejudge можна здавати розв'язки багатократно, взнавати результати кожної здачі та отримати максимальну кількість балів з усіх спроб, а без автоматичної системи все це неможливо.
2. Під час туру учасник олімпіади має право задавати запитання щодо умов задач. Задавати їх слід через систему ejudge (лінк «Надіслати питання»). Відповіді приходитимуть теж через ejudge (лінк «Уточнення»). Питання повинно бути чітко сформульоване й допускати коротку однозначну відповідь (бажано — «так» чи «ні»). Журі залишає за собою право відповідати «Без коментарів» (якщо питання нечітко поставлене, або стосується способу розв'язування, а не умови, або не стосується поточного етапу олімпіади, тощо). Як правило, журі відповідає лише автору питання, але в окремих випадках може відповідати всім учасникам. (Тому, є сенс іноді поглядувати, чи не з'явилися в єджаджі нові уточнення, навіть якщо Ви не задавали питань. Якщо вони з'являтимуться, вгорі всіх сторінок ejudge з'являтиметься також рядок на жовтому тлі з повідомленням про це.)

   Через ejudge можна повідомляти також про ситуації, які, на Вашу думку, свідчать про проблеми з ходом перевірки; журі залишає за собою право відповідати на них по суті або «без коментарів».
3. Учасник олімпіади може бути позбавлений права участі в олімпіаді:
   - через наявність на робочому місці під час туру електронних носіїв інформації (ноутбуків, планшетів, USB-накопичувачів, мобільних телефонів та будь-яких інших пристроїв з електронною пам'яттю), друкованих підручників, посібників чи довідників, попередніх записів;
   - за спробу спілкування з іншими учасниками (в т. ч. спробу передачі записок) під час туру;
   - за спробу змінити налаштування комп'ютера без прямого дозволу на те чергового вчителя;
   - за спробу використати мережу (локальну чи Internet) для будь-яких інших цілей, крім роботи з ejudge.ckipo.edu.ua;
   - за спробу здати на ejudge код, призначений не для розв'язування задачі, а для втручання у нормальний хід перевірки.
4. Щоб здавати розв'язки в автоматичну систему перевірки, слід зайти на **https://ejudge.ckipo.edu.ua** і там вибрати змагання 71 «Районна (міська) олімпіада з інформатики (програмування) 2019-20 н.р.».
5. Для входу в змагання слід використати **отримані від організаторів логіни&паролі**. Зареєструватися самостійно неможливо (запити на реєстрацію прийматимуться, але не будуть оброблені).
6. Щоб здати задачу, потрібно добитися появи маленьких прямокутничків з кодами задач A, B, C, D (наприклад, клікнувши на «Інфо») і клікнути на той, яку задачу плануєте здати.
   а) Якщо у задачі сказано «напишіть вираз», то потрібно ввести придуманий Вами вираз (сам вираз, не назву файлу) в поле навпроти слова «Відповідь» і натиснути кнопку «Надіслати!».
   б) Якщо у задачі сказано «напишіть програму», то потрібно виконати такі дії:
      - Вибрати мову програмування, яка Вам найбільше підходить, з випадного списку коло слова «Мова» (див. також деякі поради щодо мов програмування далі в цьому документі).
      - Натиснувши кнопку «Вибрати файл» (напис може залежати від браузера) навпроти напису «Файл», слід знайти і вибрати (збережений у файлову систему комп'ютера, за яким працюєте) файл з розв'язком — **вихідний (исходный, source, сирцевий)**, з розширенням **.pas, .cpp, .c, .java, .py, .cs** тощо відповідно до мови програмування; **не** файл проекту з розширенням *.sln, *.vcproj, *.iml, *.cbp, тощо (але для одно файлових програм на Delphi **.dpr**-файл одночасно є і вихідним файлом з розв'язком, і файлом проекту, тож здавати його). Натиснути кнопку «Надіслати!».
   в) Через деякий час після здачі (залежно від завантаженості сервера, від кількох секунд до кількох хвилин) з'явиться вердикт з сумою набраних балів. Для задач, в яких треба здавати програму, для не зарахованих тестів буде вказано причину не зарахування — «Неправильна відповідь», «Перевищено максимальний час роботи» (програма працювала надто довго й була обірвана єджаджем), «Помилка виконання» (програма сама по собі «злетіла» з помилкою), тощо. Для помилок компіляції можна побачити причину та номер рядка.
   г) Під час туру Вам **буде доступна інформація про те, які результати вивела Ваша програма на сервері на тесті з умови**. (Це може допомогти з'ясувати деякі технічні помилки, особливо, якщо на сервері щось працює не так, як локально.)
   д) Вміст решти тестів (крім найбільших) стане доступний через ejudge увечері 14.12.2019 або вдень 15.12.2019, а протягом самої олімпіади він є секретним.

7. Автоматична перевірка передбачає, що **<u>програма учасника чітко дотримується формату</u>** введення та виведення, **<u>вводячи й виводячи лише вказане в умові задачі</u>**. Приклади правильного дотримання формату введення/виведення наведено у пам'ятці далі в цьому документі. **<u>Виводити що б не було ще</u>** (в т. ч. й «пояснюючі фрази», як-то «writeln('Vvedit dva chysla')») **<u>не можна</u>**, бо ця «пояснююча фраза» буде сприйнята автоматичною системою перевірки за неправильну відповідь.
8. Введення/виведення можливе хоч через стандартні вхід/вихід (клавіатура/екран), хоч через файли. На вибір учасника. В т. ч., можна читати з файлу й виводити на екран. Якщо користуватися файлами — вони називаються `input.txt` та `output.txt` (в усіх задачах однаково), текстові, розміщені в поточній папці (ім'я писати без шляху). Див. також приклади наприкінці цього документу.
9. Здавати розв'язки задачі **<u>можна багато разів</u>**, і система перевірки сама вибере **<u>максимальні</u>** бали (за задачу в цілому, не за окремі тести чи блоки; якщо ж один Ваш розв'язок проходить одні тести чи блоки, інший — інші, Вам слід якось поєднати це в одній програмі).
10. Кількість спроб здачі обмежена: **<u>не більше 60 (шістдесяти) сумарно по всім задачам</u>** за тур.
11. Учасник має право скопіювати свою роботу на USB-диск чи інший носій інформації, чи відіслати собі (на email, тощо) **<u>лише після завершення туру і з прямого дозволу чергового вчителя</u>**.
12. Попередні (до апеляції) результати будуть видні на самому сайті **<u>ejudge.ckipo.edu.ua</u>**, але там будуть разом результати учасників різних районів/міст та різних класів. Про результати саме свого району/міста з розподілом по класам питайте в місцевих організаторів.

<u>Пам'ятка щодо мов програмування</u>

У випадному списку ejudge чимало мов програмування, але не всі з них мають рівні права.

Щодо мов з переліку • fpc • fpc-delphi • gcc • g++ адміністрація сервера та журі олімпіади гарантують, що вони працюють і дають можливість написати програму-розв'язок, що набирає повні бали.

Щодо мов з переліку • java • python2 • python3 адміністрація сервера та журі олімпіади гарантують, що вони працюють і дають можливість написати програму-розв'язок, що набирає значну частину балів, але не виключають, що ідейно той самий алгоритм може набирати менше балів, бо програма не вкладатиметься у обмеження часу. Такі проблеми не створюватимуться навмисно, але якщо вони виникнуть самі (внаслідок того, що ці мови за своєю природою дещо повільніші), адміністрація сервера та журі олімпіади не зобов'язані компенсовувати це.

Користуватися іншими наявними на ejudge.ckipo.edu.ua мовами (pasabc-linux, pypy, mcs, vbnc, perl, тощо) можна в тому смислі, що якщо якісь бали будуть набрані за розв'язки цими мовами, то такі бали будуть повноцінно враховані. Але ні журі олімпіади, ні адміністрація сервера не нестимуть відповідальності, ні якщо якоюсь із тих мов буде неможливо набрати повний бал, ні, навіть, якщо у тих інших мовах щось взагалі не працюватиме чи працюватиме не так. Крім того, таких мов майже точно не буде на III (обласному) етапі.

<u>Що робити, якщо я локально пишу на …</u>

<u>→ *Delphi:*</u> здавати під fpc-delphi, розуміючи, що це не справжня Delphi (якої під Linux нема). Тим не менш, fpc-delphi відтворює значну частину особливостей Delphi.

<u>→ *PascalABC:*</u> у першу чергу пробувати здавати під fpc-delphi; можна пробувати й під pasabc-linux, але він все одно відрізняється від PascalABC під Windows, а fpc-delphi хоча б швидше працює.

<u>→ *TurboPascal:*</u> як правило, здавати під fpc-delphi; але іноді виявляється доцільнішим написати на початку програми окремий рядок `{$mode tp}` і здати під fpc.

<u>Під усіма паскалями, украй небажано писати uses crt</u> Результати взаємодії `uses crt` з ejudge украй погані — програми працюють іноді правильно, іноді неправильно, іноді правильно, але довго… Причому, ejudge ніяк не повідомляє про те, що причина в `uses crt`.

<u>→ *Visual C++:*</u> здавати під g++, розуміючи, що g++ — C++, але інший. Зокрема: (1) У заголовку функції main не можна писати `tmain` та/або `TCHAR`. Можна, наприклад, `int main(int arc, char* argv[])` або `int main()` (2) Не можна мати у тексті програми `#include "stdafx.h"` та/або `#include "pch.h"`

<u>Під C/C++, не варто викликати функцію system</u> (у т. ч. й `system("pause")`).

<u>Під java, категорично не можна задавати package</u>. Інших проблем, крім цієї (та повільності виконання) зазвичай не буває, бо java — одна з найбільш кросплатформенних мов.

<u>→ *C#:*</u> здавати під mcs, розуміючи, що monoCsharp — порт під Linux досить старої версії C#, тому ситуацій, що код, який компілюється локально, не компілюється на сервері, чимало.

Пам'ятка щодо введення/виведення

Введення/виведення можливе хоч через стандартні вхід/вихід (клавіатура/екран), хоч через файли. На вибір учасника. В т. ч., можна читати з файлу й виводити на екран. Але не можна ні поперемінно читати то звідти то звідти, ні писати то туди то туди. Якщо користуватися файлами — вони називаються `input.txt` та `output.txt` (в усіх задачах однаково), **текстові**, розміщені в поточній папці (ім'я писати **без** шляху, наприклад `'input.txt'`, ні в якому разі **не** `'d:\olymp\task1\input.txt'`).

Наприклад, якби в умові задачі було сказано, що треба прочитати два цілі числа (записані в одному рядку через пробіл) і вивести їхню суму, то це можна технічно реалізувати, наприклад, так:

Мовою Паскаль, з перенаправленням на файли:

```
var a, b : longint;
begin
  assign(input, 'input.txt'); reset(input);
  assign(output, 'output.txt'); rewrite(output);
  read(a, b);
  write(a + b);
  close(output);
  close(input);
end.
```

Мовою Паскаль, без файлів:

```
var a, b : longint;
begin
  read(a, b);
  write(a + b);
end.
```

Цей (та наведений ліворуч) загальновідомі способи вже досить швидкі; саме з цим проблем у паскаля нема. Але не забуваємо, що pasabc-linux значно повільніший за fpc та fpc-delphi.

Мовою C++ з використанням файлових потоків введення/виведення:

```
#include <fstream>
#include <cstdio>
using namespace std;
int main() {
  ifstream fin("input.txt");
  ofstream fout("output.txt");
  int a, b;
  fin >> a >> b;
  fout << a+b << "\n";
  return 0;
}
```

Цей спосіб значно повільніший, ніж наведений нижче через fscanf/fprintf, але швидший, ніж наведений праворуч.

Мовою C++ з використанням потоків введення/виведення, без файлів:

```
#include <iostream>
using namespace std;
int main() {
  int a, b;
  cin >> a >> b;
  cout << a+b << endl;
  return 0;
}
```

Це дуже повільний спосіб. Головним чином тому, що cin/cout повільні, але досить істотну роль тут відіграє також endl. Просто заміна endl на "\n" сама по собі вже дає певне прискорення.

Мовою C або C++ з використанням функцій файлового введення/виведення:

```
#include <stdio.h>
int main() {
  freopen("input.txt","r",stdin);
  freopen("output.txt","w",stdout);
  int a, b;
  scanf("%d%d",&a,&b);
  printf("%d\n",a+b);
  return 0;
}
```

Зазвичай, найшвидшим є саме цей спосіб. Разом з тим, у ньому найлегше допуститися помилок.

Мовою C або C++ з використанням функцій введення/виведення, без файлів:

```
#include <stdio.h>
int main() {
  int a, b;
  scanf("%d%d",&a,&b);
  printf("%d\n",a+b);
  return 0;
}
```

Мовою Java з використанням класу Scanner та з файлами:

```
import java.io.PrintWriter;
import java.util.Scanner;
import java.io.File;
import java.io.FileNotFoundException;
public class Sum {
  public static void main(String[] args)
        throws FileNotFoundException
  {
    Scanner in = new Scanner(new File("input.txt"));
    PrintWriter out = new PrintWriter(
        new File("output.txt"));
    int a = in.nextInt();
    int b = in.nextInt();
    out.println(a + b);
    out.close();
    in.close();
  }
}
```

Це зручний, але повільний спосіб. Швидший див. нижче.

Мовою Java з використанням класу Scanner, без файлів:

```
import java.util.Scanner;
public class Sum {
  public static void
        main(String [] args)
  {
    Scanner in =
        new Scanner(System.in);
    int a = in.nextInt();
    int b = in.nextInt();
    System.out.println(a + b);
  }
}
```

Це зручний, але жахливо повільний спосіб.
Швидший (і менш зручний) спосіб наведено ліворуч нижче.

Також зверніть увагу, що у програмах мовою Java **не можна** задавати (створювати) package.

Мовою Java з використанням класу StreamTokenizer та з файлами:

```
import java.io.PrintWriter;
import java.util.Scanner;
import java.io.File;
import java.io.FileNotFoundException;
import java.io.FileReader;
import java.io.IOException;
import java.io.StreamTokenizer;
public class Sum {
  public static void main(String [] args)
        throws FileNotFoundException,
IOException
  {
    StreamTokenizer in = new
StreamTokenizer(
        new FileReader("input.txt"));
    PrintWriter out = new PrintWriter(
        new File("output.txt"));
    in.nextToken();
    int a = (int) in.nval;
    in.nextToken();
    int b = (int) in.nval;
    out.println(a + b);
    out.close();
  }
}
```

Цей незручний спосіб значно швидший, ніж через Scanner.

Мовою monoC#, з використанням Console та перенаправленням на файли

```
using System;
using System.IO;
class MainClass
{
  static void Main(string[] args) {
    Console.SetIn(File.OpenText("input.txt"));
    Console.SetOut(File.CreateText("output.txt"));
    string[] data =
            Console.ReadLine().Split();
    int a = Int32.Parse(data[0]);
    int b = Int32.Parse(data[1]);
    Console.WriteLine("{0}", a + b);
    Console.Out.Close();
    Console.In.Close();
  }
}
```

Мовою monoC#, з буферним читанням відразу всього вмісту файлу

```
using System;
using System.IO;
class MainClass
{
  static void Main(string[] args) {
    string[] allLines =
            File.ReadAllLines("input.txt");
    string[] data = allLines[0].Split();
    int a = Int32.Parse(data[0]);
    int b = Int32.Parse(data[1]);
    File.WriteAllText("output.txt", ""+(a+b));
  }
}
```

Цей спосіб значно швидший, ніж наведені праворуч та праворуч-угорі способи через Console.

Мовою monoC#, без файлів

```
using System;
class MainClass
{
  static void Main(string[] args) {
      string[] data =
            Console.ReadLine().Split();
      int a = Int32.Parse(data[0]);
      int b = Int32.Parse(data[1]);
      Console.WriteLine(a + b);
  }
}
```

Мовою Python (байдуже, 2 чи 3), з файлами:

```
inp = open("input.txt")
out = open("output.txt", "w+")

a,b = map(int, inp.readline().strip().split())
out.write(str(a+b))
```

Мовою Python (версії 2), без файлів:

```
a,b=map(int,raw_input().strip().split())
print str(a+b)
```

Мовою Python (версії 3), без файлів:

```
a, b = map(int, input().strip().split())
print(a+b)
```

Авторам пам'ятки невідомі швидші способи читання на Python. Але, наскільки відомо авторам пам'ятки, для Python основною проблемою швидкості зазвичай є не читання, а основна частина.
На ejudge.ckipo.edu.ua наявна також мова руру, котра ніби повинна бути «швидшим пітоном», але: (1) мова руру недостатньо перевірена й надається лише як додаткова, без гарантій справності; (2) наявна на ejudge.ckipo.edu.ua мова руру відповідає 2-му python, не 3-му; (3) хоча руру й помітно швидша за python, але потребує значно більше, ніж python, пам'яті, а обсяг пам'яті теж обмежений.

Можливі й технічно інші правильні способи роботи з файлами. Наведено лише **деякі з** правильних. Якщо Вам подобається інший спосіб (і Ви в ньому впевнені) — можете використовувати інший.

Усі коментарі щодо того, що одні способи швидші, інші повільніші, звісно, абсолютно неістотні саме тут, де треба прочитати лише два числа, а вивести лише одне. Але **вони стають істотними, коли обсяги введення/виведення стають справді великими,** а такі задачі в комплекті будуть.