Задачі з Основ програмування (1 семестр 2017/2018 н.р.)

Загальні бали (max 100) =
бали за семестр (max 60) + екзамен (max 40)

Бали за семестр (max 60) =
40 **домашніх задач** по 1 балу +
2 **лабораторні роботи** по 10 балів.

06.10–08.10 (ввід, вивід, змінні числових типів, присвоєння, математичні операції)

1. Написати програму, що читає з консолі введене користувачем ціле число та виводить на консоль число, вдвічи більше від введеного числа. Виконати програму покроково у середовищі Microsoft Visual Studio. При цьому переглянути значення змінних програми.

2. Написати програму для обчислення значення виразу

$$\sin\left(\frac{x+y^2+z^3}{1+(x+y+z)^2}\right)\sqrt{|x+y+z|}$$

   для заданих значеннь x, y, z типу `double`. Для обчислення функції cos скористатися методом `Math.Cos`.

3. Написати програму за обчислення площі рівнобедреної трапеці за її основами рівнобедреної трапеції і кутом при більшій основі.

09.10–12.09 (умовий оператор)

4. Написати програму для розв'язання рівнняння вигляду $ax = b$, де a, b – цілі числа, що вводяться користувачем через консоль. Програма має вивести дійсне значення x, що задовольняє рівння, якщо таке значення існує і єдине, або вивести повідомдлення про те, що рівняння не має ров'язків, або має не один розв'язок.

5. Написати програму для розв'язання рівняння вигляду $ax^2 + bx + c = 0$, де a, b, c – цілі числа, що вводяться користувачем через консоль. Програма має вивести два дійсних значення x_1, x_2, що задовольняють рівння і $x_1 < x_2$, якщо такі значення існують і єдині, або вивести повідомдлення про те, що рівняння не має дійсних розв'язків, або має один дійсний розв'язок, або має більше ніж два дійсних розв'язки.

   Для обчислення квадратного кореня дійсного не'відємного числа використати метод `Math.Sqrt` (метод стандартного класу `Math` для обчислення квадратного кореня), наприклад, рядок `Console.WriteLine(Math.Sqrt(4.0))` виводить значення 2.

6. Написати програму для обчислення суми цифр заданого натурального 3-цифрового числа.

13.09–15.09 (оператори циклу)

7. Написати програму для обчислення суми цифр у десятковому записі заданого числа типу `ulong`. Число вводиться користувачем. Програма має вивести суму цифр на консоль.

8. Написати консольну програму для моделювання руху матеріальної точки у замкненій прямокутній області на площині. Лівий нижній кут прямокутної області має координати (0,0). Матеріальна точка переміщується з часом в середині області. Вона рухається по прямій з постійною швидкістю до зіткнення зі стінкою коробки, після чого пружньо відбивається від стінки і продожує рух з тією ж швидкістю у відповідному напрямку до нового зіткнення зі стінками і т.д. Розміри області (ширина w та висота h), початкове положення матеріальної точки (координати $x_0 \in [0, w]$, $y_0 \in [0, h]$), проекції v_x, v_y її початкового вектора швидкості на осі x та y, та час t_{max} (натуральне число), до якого програма моделює рух матеріальної точки вводяться через консоль користувачем програми. Програма має вивести послідовність рядків вигляду `t x y` де `t=0,1,2,...,`t_{max} – послідовні моменти часу (від початку руху), а `x, y` – координати положення матеріальної точки в час `t`. Одиниці виміру довжин та часу обрати власний розсуд. Наприклад, програма може вивести:

```
0 1.0 1.0
1 1.1 1.5
2 1.2 2.0
```

що означає, що в початковий момент часу ($t = 0$) координати матеріальної точки – $(1, 1)$, після чого в момент часу $t = 1$ її координати – $(1.1, 1.5)$, а момент $t = t_{max} = 2$ вона досягає точки $(1.2, 3.0)$.

9*. Написати програму для наближеного обчислення значення квадратного кореня дійсного невідємного числа, введеного користувачем з консолі, не використовуючи методи класу `Math` (наприклад, `Math.Sqrt`), а користуючись операторами циклу, умовним оператором та арифметичними операціями над дійсними числами (+, –, *, /).

16.10–19.10 (інтерактивні програми)

10. Створити програму для такого сценарію. Уявімо, що є стіл, на якому лежать предмети. Користувач розповідає програмі про предмети, що є на столі (наприклад, чашка, ложка і т.п.) і їх взаємне розташування (ложка знаходиться справа / зліва / перед /... від чашки) вводячи інформацію через консоль. При введенні інформації програма запам'ятовує її. Після введення необхідних відомостей, користувач може попросити програму схематично зобразити стіл на консолі. Наприклад, якщо користувач повідомив програмі, що на столі є чашка (Ч) і ложка (Л), і що ложка знаходиться справа від чашки, і після цього попросив програму вивести схему столу на екран, то програма має зобразити таку схему: `Ч Л`.

    Взаємодію користувача з програмою організувати через вибір варіантів, наприклад:

    Програма: `Виберіть, що Ви хочете зробити`
    `(1 - розповісти про предмет на столі,`
    `2 - побачити схему столу,`
    `3 - вийти з програми):`

    Користувач: `1`

    Програма: `Виберіть тип речення, яке Ви хочете мені сказати:`
    `1 - я бачу на столі ___`
    `2 - предмет ___ знаходиться _____ від предмету _____`

    Користувач: `1`

    Програма: `Повідомте мені, який предмет, Ви бачите на столі`
    `(1 - ложку (Л), 2 - чашку (Ч)):`

    Користувач: `1`

    Програма: `ОК. Ви мені повідомили: ''я бачу на столі ложку''.`
    `Виберіть, що Ви хочете зробити далі`
    `(1 - розповісти про предмет на столі,`

```
2 - побачити схему столу,
3 - вийти з програми):
```

Користувач: 2

Програма: Схема столу:

```
Л

Виберіть, що Ви хочете зробити далі
(1 - розповісти про предмет на столі,
2 - побачити схему столу,
3 - вийти з програми):
```

Користувач: 3

Деталі щодо сенсу відношень між предметами (наприклад, чи слово "зліва" означає "безпосередньо зліва" і т.п.) продумати самостійно.

20.10–22.10 (масиви)

11. Написати програму, що забезпечує введення з консолі послідовності з не більше 100 цілих чисел та знаходження найбільшого серед них. Зберігати введені числа в масиві.

12. Написати програму для знаходження трикутника з найбільшим можливим периметром, вершини якого належать заданій користувачем з консолі послідовності з попарно різних 10 точок на площині, заданих цілочисельними координатами x, y. Якщо такого трикутника не існує (усі точки розташовані на одній прямій), програма має вивести відповідне повідомлення.

13. Написати програму для транспонування прямокутної матриці цілих чисел розміру не більше 50×50. Розмір та елементи матриці вводяться через консоль, після чого транспонована матриця виводиться програмою на консоль. Деталі способу введення та форму виведення даних продумати самостійно.

    Наприклад, при введенні матриці

    $$\begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{pmatrix},$$

    програма має вивести матрицю

    $$\begin{pmatrix} 1 & 4 \\ 2 & 5 \\ 3 & 6 \end{pmatrix}.$$

14*. Написати програму знаходження кількості зв'язних областей у чорно-білому зображенні розміру не більше 20×20, заданому двовимірним масивом чисел 0, 1, що вводиться користувачем через консоль.

Наприклад, при вводі зображення

```
0 0 1 0 0 0 0 0 0
0 1 1 1 0 0 0 0 0
1 1 1 1 1 0 1 1 1
0 0 0 0 0 0 1 1 1
0 0 0 0 0 0 0 0 0
```

програма має виводити `2`, тобто зображення містить дві зв'язні області (трикутник зліва та прямокутник справа).

23.10-26.10 (прості структури в C#)

15. Написати програму для додавання і віднімання грошових сум. Програма повинна надати можливість користувачу програми ввести дві грошові суми, кожна з яких вводиться як два невід'ємних цілих числа – кількість гривень і кількість копійок в межах від 0 до 99, та повинна вивести суму цих двох грошових сум та різницю між більшою з них і меншою (або 0 грн 0 коп, якщо вони однакові) у вигляді двох пар невід'ємних цілих чисел – кількості гривень і кількості копійок в межах від 0 до 99.

Наприклад, якщо ввести суми 1 грн 20 коп та 2 грн 10 коп, програма має вивести у якості суми цих двох грошових сум 3 грн 30 коп, а у якості різниці між більшою і меншою сумою – 0 грн 90 коп.

У програмі подати грошові суми екземплярами структури `Money`, наведеної нижче, з двома цілочесельними полями – `grn` (кількість гривень) і `kop` (кількість копійок в межах від 0 до 99).

```
public struct Money
{
    public uint grn;
    public byte kop;
}
```

16. Написати програму для визначення типу трикутника (прямокутний, рівнобедрений, і т.п.) заданого координатами x, y його вершин на пощині. Координами трьох вершин вводиться користуачем з консолі у вигляді пар цілих чисел. Програма має вивести на консоль тип трикутника (напр. "Прямокутний"), або

повідомлення про те, що введені точки не утворюють трикутник (лежать на одній прямій).

У програмі подати введені користувачем точки екземплярами структури `Point`, наведеної нижче:

```
public struct Point
{
    public int x;
    public int y;
}
```

17*. Написати програму для визначення, чи задана послідовність з не більше 100 точок (подана парами цілих чисел – координат x, y на площині) є переліком вершин простого многокутника (без самоперетинів). Прогрма має надавати користувачу можливість ввести точки з консолі і має виводити на консоль повідомлення `Так` чи `Ні`. У програмі подати введені точки масивом, елементами якого є екземпляри структури з двома полями (координатами точки).

27.10–29.10 (символи і рядки)

18. Написати програму для обчислення арифметичного виразу вигляду `число операція число`, де число є натуральним числом, операція – одна з $+$, $-$, $*$, $/$. Вираз вводиться через консоль користувачем. Числа від операції відділені однією або більше прогалин. Програма має вивести значення виразу.

30.10–02.11 (ділення навпіл, бінарний пошук)

19. Написати програму, яка за заданими користувачем цілим числом n в діапазоні 1...20 та числом $a \in \{2, 3\}$ обчислює a^n (не користуючись методами класу `Math`), при цьому виконуючи не більше $2\log(n)$ операцій множення.

    Наприклад, a^7 можна обчислити, виконавши $4 < 2\log_2(7)$ операції множення:

```
b = a * a
c = b * b
result = a * b * c
```

20. Написати програму, яка за заданим масивом A з N дійсних чисел та дійсним числом S визначає, чи існують натуральні числа i, j в діапазоні $1...N$, такі, що $A[i] + A[j] = S$.

21. Інверсією в масиві A з N цілих чисел назвемо таку пару натуральних чисел (i, j), що $1 \le i \le j \le N$ та $A[i] > A[j]$. Написати програму, яка визначає кількість інверсій в заданому масиві A з N цілих чисел.

03.11–07.11 (довга арифметика)

22. Написати програму, яка додає два натуральні числа, задані в двійковій системі числення масивами цифр. Кількість цифр в двійковому записі кожного з вхідних чисел не перевищує 10000.

23. Написати програму, яка знаходить добуток двох натуральних чисел, заданих в двійковій системі числення масивами цифр. Кількість цифр в двійковому записі кожного з вхідних чисел не перевищує 10000.

24. Написати програму для обчислення цілої частини та остачі від ділення невід'ємного числа M в діапазоні $1...10^{10000}-1$, заданого десятковим записом у вигляді рядка з не більше ніж 10000 цифр на ціле число в діапазоні від $1...2^{30}$.

08.11–12.11 (довга арифметика 2)

25. Написати програму для переведення натурального числа, заданого масивом цифр у двійковій системі числення в десяткову систему. Кількість цифр в двійковому записі вхідного числа не перевищує 10000.

26. Написати програму для переведення натурального числа, заданого масивом цифр у десятковій системи числення в двійкову систему. Кількість цифр в десятковому записі вхідного числа не перевищує 10000.

13.11–14.11 (теорія чисел)

27. Написати програму для виведення подання заданого цілого числа N в діапазоні $1...2^{30}$ у вигляді добутку простих чисел.

28. Написати програму для знаходження найбільшого спільного дільника та найменшого спільного кратного двох цілих чисел в діапазоні $1...2^{30}$. Скористатись алгоритмом Евкліда.

29. Написати програму для виведення усіх простих чисел в межах від 1 до заданого числа N в діапазоні $1...2^{30}$.

15.11–19.11 (динамічне програмування)

30. Написати програму для обчислення n-го члена (F_n) послідовності Фібоначчі, визначеної таким чином: $F_0 = 1$, $F_1 = 1$, $F_n = F_{n-1} + F_{n-2}$, $n \geq 2$. Число n в діапазоні 1...100 вводиться у якості вхідних даних програми. Програма має вивести число F_n.

31. Написати програму для виводу усіх способів подання заданого натурального числа N в діапазоні $1...2^{30}$ у вигляді суми неспадаючої послідовності натуральних чисел.

32. Написати програму, яка за двома заданими рядками довжини не більше 25, що складаються з символів A-Z, визначає кількість способів, якими можна отримати перший рядок з другого шляхом викреслення деяких символів. Наприклад, якщо задано рядки `ab` та `abcab`, то перший рядок можна отримати з другого двома способами: або викресленням перших трьох символів, або останніх трьох символів. Якщо задано рядки `abcab` та `ab`, то другий рядок не можна отримати з першого шляхом викреслення символів, тому в такому випадку відповідь програми має бути нуль.

20.11–21.11 (перебір)

33. Написати програму для визначення кількості способів, якими можна подати задане ціле число N у вигляді суми строго зростаючої послідовності натуральних чисел.

34. Написати програму, яка для заданого користувачем рядка з не більше ніж 6 символів виводить усі рядки, що можуть бути отримані із заданого рядка шляхом перестановки деяких символів. Наприклад, якщо користувач задав програмі рядок `abc`, то програма має вивести такі рядки (у довільній послідовності): `abc`, `acb`, `bac`, `bca`, `cab`, `cba`.

35. Написати програму, яка за заданими натуральними числами n, x та a_1, a_2, ..., a_n знаходить один з способів розстановки знаків "+" або "-" між числами послідовності a_1, a_2, ..., a_n, так, щоб значення отриманого виразу було рівне x (якщо такий спосіб існує).

Якщо вказаного способу розстановки знаків не існує, програма має виводити відповідне повідомлення.

22.11–26.11 (різне)

36. Написати програму для сортування за неспаданням заданої користувачем послідовності з N ($2 \leq N \leq 100$) цілих чисел. Програма отримує на вхід число N та N цілих чисел та виводить на консоль вказані числа, впорядковані за неспаданням.

37. Написати програму обчислення площі простого n-кутника ($n \leq 100$) на площині, заданого послідовністю n пар цілих чисел – координат x, y своїх вершин у порядку одного з варіантів їх обходу.

38. Написати програму, яка за заданим користувачем числом N в межах $1...2^{30}$ знаходить кількість різних прямокутників з цілочисельними довжинами сторін, які мають площу N.

27.11–30.11 (різне 2)

39. Написати програму, яка за заданою користувачем квадратною таблиця розміру $N \times N$ (N в діапазоні 2...20), у клітинках якої знаходяться цифри 0-9, знаходить і виводить шлях з лівої нижньої у праву верхню клітинку таблиці, який є послідовністю клітинок, в якій кожна наступна клітинка знаходиться або безпосередньо над, або безпосередньо справа від попеденьої, для якого сума цифр у клітинках вздовж шляху є мінімально можливою (якщо таких шляхів більше одного, програмі достатньо знайти лише один з них). Наприклад, якщо користувач вводить таблицю

```
0 0 1
1 2 1
0 0 1
```

то програма може вивести знайдений шлях у такому вигляді (зірочами відмічено клітини, що належать шляху):

```
* * *
*
*
```

При цьому сума цифр у клітинках такого шляху є 2 і є мінімально можливою серед усіх шляхів з лівої нижньої у праву верхню

клітинку таблиці, в яких кожна наступна клітинка знаходиться або безпосередньо над, або безпосередньо справа від попеденьої.

40. Написати програму, яка визначає, чи можна отримати заданий користувачем рядок, що складається з символів [,], (,) шляхом виконання послідовності таких дій:

    а) заключення порожнього або одного з вже отриманих рядків у круглі, квадратні, або фігурні дужки;

    б) зчеплення двох вже отриманих рядків (наприклад, з () і [] отримується ()[]).

    Програма має виводити повідомлення `Так` або `Ні`. Наприклад, рядок [[]()[]][] можна отримати виконанням дій а) та б): можна отримати () та [] за допомогою операції а), з них за попомогою б) можна отримати []()[], звідки за допомогою операції а) можна отримати [[]()[]], а з [[]()[]] і [] можна отримати [[]()[]][] за допомогою операції б). Тому для вхідного рядка [[]()[]][] програма має вивести `Так`. Але для рядка (] програма має вивести `Ні`.

01.12–08.12: доскладання завдань, що не були виконані раніше, підготовка до екзамену

30.10–08.12: лабораторна робота №1 (10 балів)

Написати програму мовою C#, вихідний код якої складається з єдиного файлу, що містить рівно один рядок, таку, що, при запуску програма виводить на консоль єдиний рядок свого вихідного коду у зворотньому порядку – з останнього символу до першого, та завершує виконання.

30.10–08.12: лабораторна робота №2 (10 балів)

Дана консольна програма на мові C# (див. нижче), яка при запуску запитує ім'я користувача (англ. user name) та кодове слово (англ. code word). В залежності від введеного імені користувача та кодового слова, програма може вивести деякий рядок символів на консоль і успішно завершитися, або аварійно завершитися, або "зависнути".

Завдання:

1) Ввести, скомпілювати за запустити наведену нижче програму на комп'ютері.

2) Підібрати таке кодове слово, щоб при вводі імені студента латинськими літерами у якості імені користувача та цього кодового слова наведена нижче програма виводила на консоль повідомлення `OK` та успішно завершувалась.

3) Написати програму на мові C# , що дозволить для будь-якого заданого імені користувача, що складається з не більше ніж 5 алфавітно-цифрових символів (0-9, a-z, A-Z) знайти одне з можливих кодових слів, таке, щоб наведена нижче програма при вводі цього імені користувача та кодового слова виводила повідомлення `OK` та успішно завершувалась.

```
using System;

class Program
{
    public struct Proposition
    {
        public bool IsAssertion;
        public string Operation;
        public int Arg1;
        public int Arg2;
    }

    static void Main()
    {
        Console.WriteLine("User name: ");
        string name = Console.ReadLine();
        Console.WriteLine("Enter code word: ");
        string codeword = Console.ReadLine();
        Console.WriteLine();
        int i = 0; byte c1 = 0, c2 = 0;
        for (i = 0; i < name.Length; i++)
            c1 += (byte)name[i];
        for (i = 0; i < codeword.Length; i++)
            c2 += (byte)codeword[i];
        if (c1 != c2)
            Console.WriteLine("Wrong code word");
        else
        {
            codeword = "bakbbkbckbbbambcbbmbcbambdtbetbbjbfu" + codeword;
            byte[] s = new byte[256];
            byte top = (byte)(s.Length - 1);
            s[--top] = 0;
            Proposition[] proof = new Proposition[256];
            bool cheat_mode = true;
            int pc = 0, goal = -1;
            for (i = 0; i < codeword.Length; i++)
            {
                int code = codeword[i] - 'a';
```

```
if (!cheat_mode)
{
    if ((codeword[i] >= '0' && codeword[i] <= '9') || (
       codeword[i] >= 'A' && codeword[i] <= 'Z'))
        code = codeword[i] ^ c1;
    else if (((code ^ c1) >= '0' && (code ^ c1) <= '9') ||
        ((code ^ c1) >= 'A' && (code ^ c1) <= 'Z'))
        break;
}
int A, B; if (i == 0) cheat_mode = true;
switch (code)
{
    case 0: i = codeword.Length; break;
    case 1: byte val = (byte)(codeword[++i] - 'a');
        if (!cheat_mode)
        {
            if ((codeword[i] >= '0' && codeword[i] <= '9')
                || (codeword[i] >= 'A' && codeword[i] <=
               'Z'))
                val = (byte)(codeword[i] ^ c1);
        }
        s[--top] = val;
        break;
    case 2: top++; break;
    case 3: s[++top] += s[top - 1]; break;
    case 4: s[++top] -= s[top - 1]; break;
    case 5: s[++top] *= s[top - 1]; break;
    case 6: s[++top] /= s[top - 1]; break;
    case 7: s[--top] = s[top + 1]; break;
    case 8:
        if (goal < 0 || proof[goal].IsAssertion)
            Console.Write((char)(s[top++] + 'A'));
        break;
    case 9:
        int target = s[top++], value = s[top++];
        if (target > 127) target = target - 256;
        if (value != 0)
            i += target;
        break;
    case 10:
        proof[pc].IsAssertion = false;
        proof[pc].Operation = s[top++].ToString();
        pc++;
        break;
    case 11:
        proof[pc].IsAssertion = false;
        proof[pc].Operation = "NOT";
        proof[pc].Arg1 = s[top++];
        pc++;
        break;
    case 12:
        proof[pc].IsAssertion = false;
        proof[pc].Operation = "IMPLICATION";
        proof[pc].Arg1 = s[top++];
        proof[pc].Arg2 = s[top++];
        pc++;
        break;
    case 13:
    case 14:
```

```
            A = s[top++]; B = s[top++];
            int[] x = new int[256], y = new int[256];
            x[0] = A; y[0] = B;
            int count = 1;
            while (count > 0)
            {
                count--;
                if (proof[x[count]].Operation != proof[y[count
                   ]].Operation)
                {
                    count = -1;
                    break;
                }
                else if (proof[x[count]].Operation == "NOT")
                {
                    x[count] = proof[x[count]].Arg1; y[count]
                       = proof[y[count]].Arg1; count++;
                }
                else if (proof[x[count]].Operation == "
                   IMPLICATION")
                {
                    int xc = x[count], yc = y[count];
                    x[count] = proof[xc].Arg1; y[count] =
                       proof[yc].Arg1; count++;
                    x[count] = proof[xc].Arg2; y[count] =
                       proof[yc].Arg2; count++;
                }
            }
            if (count == 0)
            {
                if (code == 13)
                {
                    proof[pc].IsAssertion = true;
                    proof[pc].Operation = "IMPLICATION";
                    proof[pc].Arg1 = A;
                    proof[pc].Arg2 = B;
                    pc++;
                }
                else if (proof[A].IsAssertion || proof[B].
                   IsAssertion)
                {
                    proof[A].IsAssertion = true;
                    proof[B].IsAssertion = true;
                }
            }
            break;
        case 15: ;
            A = s[top++]; B = s[top++];
            if (proof[B].Operation == "IMPLICATION" && proof[B
               ].Arg2 == A)
            {
                proof[pc].IsAssertion = true;
                proof[pc].Operation = "IMPLICATION";
                proof[pc].Arg1 = A;
                proof[pc].Arg2 = B;
                pc++;
            }
            break;
        case 16:
```

```
                        A = s[top++]; B = s[top++];
                        if (proof[A].Operation == "IMPLICATION" && proof[B
                            ].Operation == "IMPLICATION" &&
                            proof[proof[A].Arg2].Operation == "IMPLICATION
                                " &&
                            proof[proof[B].Arg1].Operation == "IMPLICATION
                                " &&
                            proof[proof[B].Arg1].Arg1 == proof[A].Arg1 &&
                            proof[proof[B].Arg1].Arg2 == proof[proof[A].
                                Arg2].Arg1 &&
                            proof[proof[B].Arg2].Operation == "IMPLICATION
                                " &&
                            proof[proof[B].Arg2].Arg1 == proof[A].Arg1 &&
                            proof[proof[B].Arg2].Arg2 == proof[proof[A].
                                Arg2].Arg2)
                        {
                            proof[pc].IsAssertion = true;
                            proof[pc].Operation = "IMPLICATION";
                            proof[pc].Arg1 = A;
                            proof[pc].Arg2 = B;
                            pc++;
                        }
                        break;
                    case 17:
                        A = s[top++]; B = s[top++];
                        if (proof[A].Operation == "IMPLICATION" && proof[B
                            ].Operation == "IMPLICATION" &&
                            proof[proof[A].Arg1].Operation == "NOT" &&
                                proof[proof[A].Arg2].Operation == "NOT" &&
                            proof[proof[A].Arg1].Arg1 == proof[B].Arg2 &&
                            proof[proof[A].Arg2].Arg1 == proof[B].Arg1)
                        {
                            proof[pc].IsAssertion = true;
                            proof[pc].Operation = "IMPLICATION";
                            proof[pc].Arg1 = A;
                            proof[pc].Arg2 = B;
                            pc++;
                        }
                        break;
                    case 18:
                        A = s[top++];
                        if (proof[A].Operation == "IMPLICATION" &&
                            proof[A].IsAssertion &&
                            proof[proof[A].Arg1].IsAssertion)
                            proof[proof[A].Arg2].IsAssertion = true;
                        break;
                    case 19:
                        if (cheat_mode)
                            proof[s[top++]].IsAssertion = true;
                        break;
                    case 20:
                        if (cheat_mode) goal = s[top++];
                        cheat_mode = false;
                        break;
                }
            }
        }
    }
}
```