Први контролни задатак, група Б (решења) 21.10.2020.

1.

2. Sve izjave su netačne osim izjave 8, 10

1. ASCII (Ansi Standard Code for **Information Interchange**)
zadat je u formatu **UREĐENE** liste karaktera koje su predstavljenih svojim karakterskim kodovima.
2. 7-bitan
3. Prosiren ASCII ima 8-bitnu reprezentaciju ASCII karaktera, pri čemu je H.O. bit postavljen na 0 (umesto izvornog 7-bitnog koda).
4. ASCII zapis se jednostavno deli u 4 grupe korišćenjem bitova 5 i 6
5. Kod blanko znaka manji je od koda ma kog koda slova, cifre
6. Kodovi cifara uređeni su u uzastopnom RASTUĆEM redosledu.
7. Zato kada je zadovoljena nejednakost: kod('0')<=kod(znak)<=kod('9')
znamo da je znak cifra
9. Kodovi velikih slova latinice A..Z (26 slova) uređeni su saglasno engleskoj abecedi i takođe se ređaju bez razmaka.

3. Code::Blocks је обједињено развојно окружење чију структуру чине:
Едитор изворног програмског кода (са специјалним функционалностима: syntax highlighting, autocomplete, аутоматско форматирање кода,...)
Преводилац (engl. compiler)
Повезивач (engl. linker)
Дебагер (engl. debugger)
Менаджер датотека (енгл. File manager)

4.
Основне C ili C++ (iostream, stdio) улазно/излазне операције потичу са конзоле или су усмерене као конзоли.
Конзолне апликације представљају врсту апликација са којима корисници врше интеракцију користећи конзолни прозор у оквиру којег уносима са тастатуре контролишу ток извршавања саме апликације и читају исписе обраде које овае апликације дају за резултат. Конзолне апликације нису апликације са богатим графичким интерфејсом (desktop, web, mobilne aplikacije,...) и на различитим уређајима.

Подсетите се:
http://icarus.cs.weber.edu/~dab/cs1410/textbook/1.Basics/io.html

5. Број прегледа видео-клипа је постао већи од највећег означеног броја који може да се смести у 32-битни регистар који је YouTube тада користио за бројач.

6. У овом задатку је било довољно приметити да двоцифрени завршетак мора бити једнак 25.

Некоректан приступ: коришћење библиотечке функције pow и њено имплицитно кастовање у ужи тип int. Ученици који су користили овај приступ су већ на почетку уочили да могу освојити укупно 1 тест пример са овим решењем.

```
#include <iostream>
using namespace std;
int main()
{ long long n;
   cin>>n;
   cout<<25;
   return 0;
}
```

7.
Дакле, број флаша се одређује као цео део количника количине лимунаде и запремине флаше. Основна формула се добија ако користимо највећи цео број који није већи од датог броја.

Користећи изведену формулу директно, на примеру датом у поставци задатка добијамо резулатат 2 уместо 3. То је зато што је резултат дељења 0.6 / 0.2 једнак 2.9999999999999996, који кад се заокружи наниже даје 2. То је био један од разлога зашто једном применом дељења без обраде тачности резултата ученици освајају само неке тест примере.

Овај проблем ћемо избећи ако количнику L/F:

1. додамо мали позитиван број пре заокруживања или
2. обрадимо унесене вредности F,L и припремимо их да обављамо дељење у задатој тачности (по броју децималних места).

Приступ 1: Остаје да видимо колики тај број треба да буде.
Најмања вредност количника L/F коју уопште можемо добити је количник најмање количине лимунаде и највеће запремине флаше, дакле 0.001 / 5 = 0.0002. То значи да се флаше пуне у корацима (“квантима”) од по најмање 0.0002 флаше. Према томе, додавањем броја eps довољно мањег од 0.0002 нашем количнику (на пример eps=0.000001=1e−6), нећемо пребацити следећи цео број, то јест такво додавање неће покварити резултат када је он тачан. Са друге стране, ако је резултат требало да буде цео број, али је због ограничене тачности реалних бројева мањи од целог за рецимо 1e-14 (или мање), додавањем eps ћемо управо пребацити следећи цео број и тиме исправити грешку.

Приступ 2:

```
#include<iostream>
using namespace std;

double a, b;
int main() {
cin >> F >> L;
int aa=(int)(F*(double)1000);
int bb=(int)(L*(double)1000);
cout << bb/aa;
return 0;
}
```

8.
Класичан поступак конверзије подразумева да се израчуна колико у једном јарду има инча ($3 \cdot 12 = 36$) и да се тај број помножи бројем јарди, да се број инча у једној стопи (12) помножи бројем стопа и на крају да се на збир ова два броја дода број инча. Тиме добијамо формулу $j \cdot 36 + s \cdot 12 + i$

```
#include<iostream>
using namespace std;

int main() {
int j, s, i;
cin >> j >> s >> i;
cout << j*36+s*12+i;
return 0;
}
```

9.
Задатак може бити решен грубом силом, тј. испробавањем свих могућности расподеле додатне дужине. За сваку дужину i између 0 и c, додајемо i страници a и c−i страници b, израчунавамо површину и тражимо максимум тако добијених површина.

Ово решење не осваја све поене, јер не задовољава временско ограничење. Зато се ефикасност мора побољшати.

Од свих правоугаоника датог фиксираног обима, највећу површину има квадрат. Заиста, ако је познат обим правоугаоника $O=2(a+b)$, тада је познат и полуобим $a+b=s$.
Површина $a \cdot b=a \cdot (s-a)=a \cdot s-a \cdot a=s^2/4-(a-s/2)^2$. Пошто је $(a-s/2)^2 \geq 0$, површина не може бити већа од $s^2/4$, а једнака је тој вредности када је $a=b=s/2$. Зато увећање треба направити тако да се добије облик који је што сличнији квадрату.

Нека је $a \leq b$. Ако је $a+c \leq b$, тада се целокупан износ увећања c може додати на мању страницу a. У супротном се прво краћа страница a продужи тако да постане једнака дужој страници b, а затим се преостали износ увећања $(c-(b-a))$ подели што равномерније могуће (ако је то паран број може се добити квадрат, а ако није, тада се добија правоугаоник код којег је једна страница за један дужа од друге). У имплементацији тај ефекат можемо постићи тако што страницу b увећамо за $\lfloor c-(b-a)/2 \rfloor$ и $\lceil c-(b-a)/2 \rceil$.

```
#include <iostream>
#include <algorithm>

using namespace std;

int main() {
long long a, b, c;
cin >> a >> b >> c;
if (a > b) swap(a, b);
if (c <= b - a)
a += c;
else {
long long preostalo = c - (b - a);
a = b + preostalo / 2;
b = b + (preostalo + 1) / 2;
}
cout << a * b << endl;
return 0;
}
```

Скала оцена

95 - 80 5
79 - 62 4
61 – 45 3
44 – 30 2