Част 1. Програмиране

1.1. Формат на състезанието в секция „Програмиране“

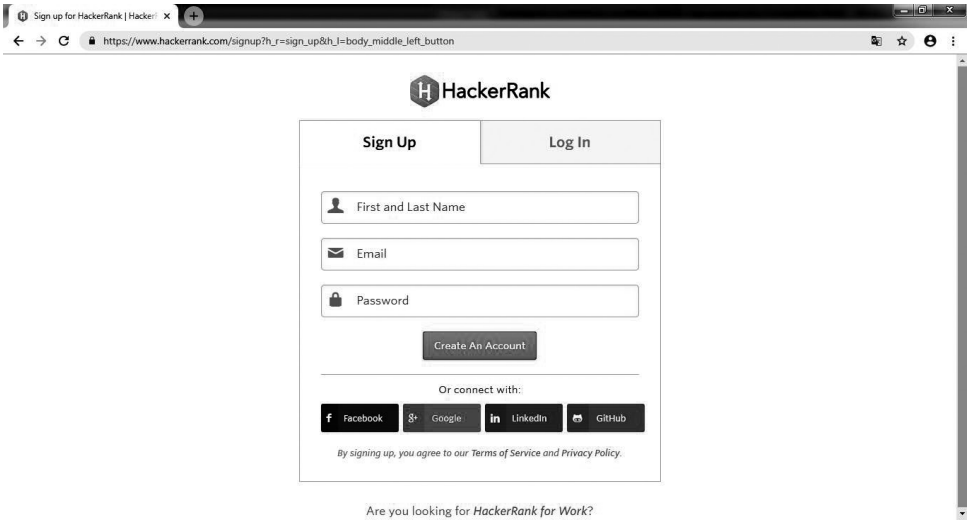
Целта на състезанието в секция „Програмиране“ е да се предостави възможност за творческа изява на учениците от средните училища в областта на програмирането.

Участниците в състезанието работят самостоятелно върху 5 задачи, публикувани в платформата HackerRank[1].

HackerRank е платформа, в която програмисти от целия свят могат да решават задачи от различни области в сферата на компютърните науки: алгоритми, машинно обучение, изкуствен интелект, бази от данни, криптография и сигурност и др., както и да практикуват функционално програмиране.

Ученическото състезание се създава в платформата, като на участниците се съобщава интернет адреса за достъп до него в деня на провеждане на състезанието.

Предварително е необходимо всички участници да имат активна регистрация в HackerRank (фиг. 1.1).



Фиг. 1.1. Форма за регистрация в платформата HackerRank

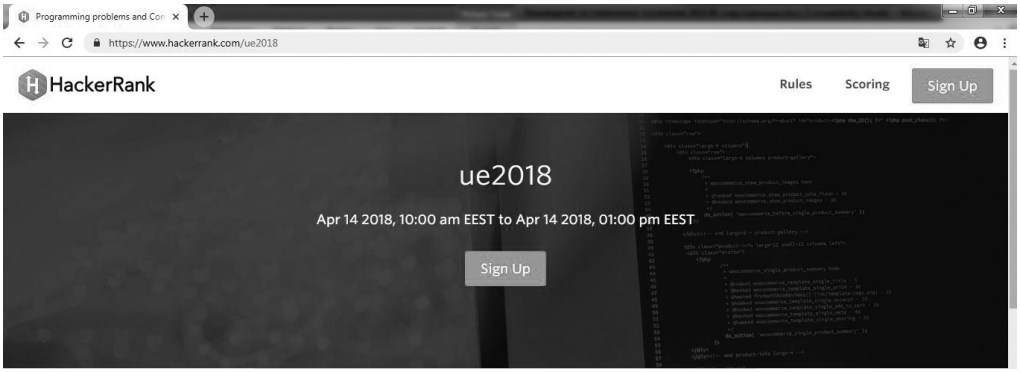
При регистрация задължително се попълват имената на участника, валиден имейл и парола. На посочения имейл се получава връзка за активиране на регис-

[1] HackerRank, <https://www.hackerrank.com> (1.09.2018).

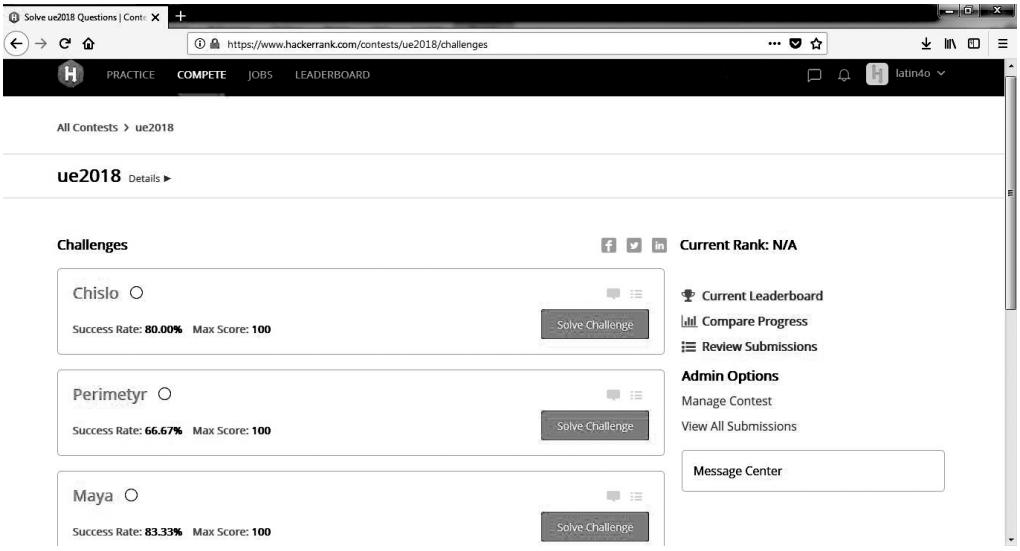
трацията.

В платформата има богата база със задачи от различни области, като могат да се преглеждат и изпратените решения от други потребители.

На фиг. 1.2 е представена началната страница на вече проведено състезание. В средата на страницата, както и в горната дясна част, има бутон *Enter Contest*, от който се преминава към списъка със задачи, заложени в състезанието (фиг. 1.3).

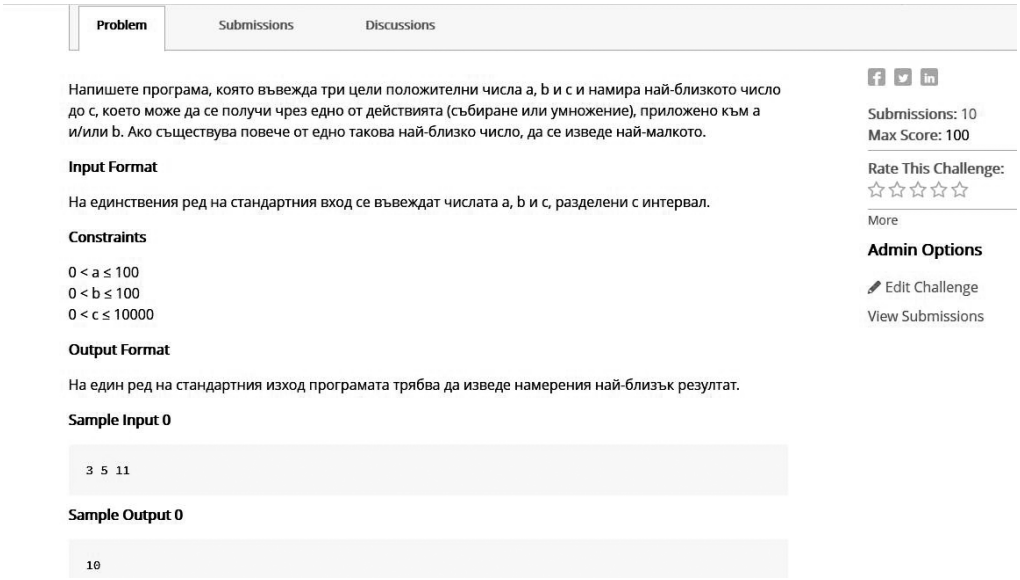


Фиг. 1.2. Начална страница на състезание ue2018



Фиг. 1.3. Страница със задачи от състезание ue2018

След натискане на бутон *Solve Challenge* се отваря страница с условието на избраната задача (фиг. 1.4).



Фиг. 1.4. Страница с описание на задача в HackerRank

Всяка страница с конкретна задача съдържа вграден онлайн редактор, в който може да се пише и да се тества кодът (фиг. 1.5). Ако участникът предпочита да работи с платформа като Visual Studio, NetBeans, Eclipse, може да го направи и след това да копира кода в онлайн редактора в страницата на задачата, за да тества и за да изпраща решението.

Фиг. 1.5. Онлайн редактор на страницата на избрана задача

Независимо от избрания подход за работа, когато участникът има вариант на решението, може да го изпълни от бутона *Run Code* (фиг. 1.5). Решението се

теста със заложените за конкретната задача и наличните в условието тестови примери. Това обаче не е реално предаване на решението на задачата.

Реалното изпращане на решението на задачата се осъществява при натискане на бутон *Submit Code*. Решенията на задачите се проверяват с неизвестни на състезателите тестови данни, едни и същи за всички участници, подготвени предварително от организационния комитет. За целия комплект от тестове е предвиден лимит от време, който не е известен на състезателите. За правилно се признава решение, което намира верен резултат за всички тестови примери в рамките на определения лимит от време.

Съобщенията, които платформата извежда при тестването на изпратеното решение, са:

- *Accepted* – кодът е преминал успешно всички тестови примери.
- *Wrong Answer* – резултатът от кода не отговаря на заложения за съответния тестови пример.
- *Terminated due to timeout* – кодът не работи достатъчно ефективно. Необходимо е оптимизиране на алгоритъма.
- *Runtime error/Segmentation Fault* – кодът приключва аварийно. Грешката може да се дължи например на опит за деление на 0, излизане извън обхвата на даден масив и т.н.
- *Abort Called* – използват се прекалено много ресурси. Вероятно създаденият масив е прекалено голям и надхвърля границите на паметта.

След предаване на решението чрез позициониране върху иконата на всеки тестови пример участникът може да види времето за изпълнение на тестовия пример и резултата.

При неверен резултат за един или няколко от тестовите примери решението на задачата не се приема за вярно и участникът може да коригира кода и да изпрати ново решение на същата задача, като за това има неограничен брой опити.

Класирането се извършва по броя на правилно решените задачи. При равен брой решени задачи за всяка правилно решена се пресмята времето (в минути) от началото на състезанието до момента на предаването на отговора ѝ. Сумират се получените времена и сумата се увеличава с по 20 минути за всяко неправилно решение. Участникът с по-малка сума на така изчислените времена се класира на по-предна позиция.

Всеки състезател разполага с един персонален компютър. Официалните езици за програмиране по време на състезанието са Java и C/C++. Времето за работа е 180 минути. Резултатът на всеки участник се визуализира в онлайн платформата.

Езикът за програмиране C++ е един от най-популярните програмни езици. Той се появява през 1980 г. на база на достиженията на езика C, като запазва синтаксиса му и предлага нови средства за обектноориентирано програмиране.

Въвеждат се редица подобрения в типовете данни, както и нови библиотечни функции.

Основните характеристики на езика C++ са:

• преносимост на програмите, написани на C++, поради машинната независимост на езика;

• компактност на програмите, като конструкциите на езика се реализират с 27 ключови думи (оператори);

• ефективен механизъм за връзка с основните програмни единици – функциите;

• рекурсия;

• библиотеки от стандартни функции;

• указатели;

• класове и обекти, чрез които се реализира обектноориентираното програмиране;

• приложение в широк спектър от области.

Java е съвременен език за програмиране от високо ниво с отворен код, лесен за научаване, което го прави и подходящ за начинаещи. Освен това е и широко разпространен, масово използван при разработката на сървърни системи и мобилни приложения; разполага с многобройни библиотеки и технологични рамки и съответно дава много перспективи за работа и професионално развитие. Java комбинира парадигмите на процедурното, обектноориентираното и функционалното програмиране с лесен за употреба синтаксис. Кодът, написан на Java, не се компилира до машинен код за определен процесор, а до специфичен за езика код, наречен байт код (Java bytecode). Поради това за изпълнението на програма, написана на Java, е необходима виртуална машина – Java Virtual Machine (JVM)[2].

C/C++ и Java са официалните езици и на Републиканската олимпиада по програмиране, която се провежда всяка година.

1.2. Задачи от състезанието през 2018 г.

Следващите задачи са включени в първото по рода си Национално ученическо състезание по информатика за ученици от XI и XII клас, проведено на 5 февруари 2018 г. в Икономически университет – Варна.

[2] Наков, Св. Основи на програмирането с Java. <https://java-book.softuni.bg/> (20.07.2018).

Задача 1. ЗНАЙКО[3]

Автор: Зорница Дженкова

Знайко намислил едно цяло число. От него извадил числото `a` и най-голямото трицифрено число, което се дели на `b`. Така Знайко получил най-малкото четирицифрено число, което се дели на `c`. Да се напише програма *znayko*, която намира числото, което е намислил Знайко?

Вход

На стандартния вход се въвеждат три цели числа a, b и c, отделени с по един интервал.

Изход

На стандартния изход да се изведе едно цяло число – намисленото от Знайко.

Ограничения

$1 \leq a \leq 1000$
$1 \leq b < 1000$
$1 \leq c < 1000$

Пример

Вход

3 6 9

Изход

2007

Авторско решение

```
#include <iostream>
using namespace std;
int main()
{
    int a,b,c,x;
    int i,j;
    cin>>a>>b>>c;
    int max,min;
    for(i=999;i>=100;i--)
        if(i%b==0)
        {
           max=i;
           break;
        }
    for(j=1000;j<=9999;j++)
       if(j%c==0)
       {
           min=j;
```

[3] Зимни състезания по информатика, Велико Търново, 10 – 12 февруари 2012 г.

```
            break;
        }
    x = a + max + min;
    cout<<x<<endl;
}
```

Анализ на решението

С оператор за цикъл, при който броячът намалява, се намира най-голямото трицифрено число, което се дели на `b`. За начало на цикъла се взема 999 (най-голямото трицифрено число), а за край – 100 (най-малкото трицифрено число). В тялото на цикъла се използват условен оператор и операция деление с остатък (%). Когато се намери първото число, което се дели на `b`, се излиза от цикъла с оператор `break`. Резултатът се записва в променлива `max`.

Аналогично се намира най-малкото четирицифрено число, което се дели на `c`. Управляващата променлива на цикъла започва от 1000 и се увеличава до 9999. Резултатът се записва в променливата `min`.

Да се означи с `x` числото, което е намислил Знайко. Тогава може да се запише:

```
x – a – max = min
```

Следователно `x` ще се пресметне по следния начин:

```
x = a + max + min
```

Задача 2. ПРОСТИ ЧИСЛА[4]

Талантливчо Информатиков, който скоро научил от часовете по математика кое число наричаме просто, започнал да пише на лист последователност от числа, като искал да преброи колко са простите числа в тази редица. Помогнете на Талантливчо, като напишете програма *prime*, която намира този брой.

Вход

От първия ред на стандартния вход се въвежда броят n на числата в редицата.

От втория ред на стандартния вход се въвеждат дадените n цели положителни числа (представляващи елементите на редицата), разделени с интервали.

Изход

На един ред на стандартния изход програмата трябва да изведе търсения брой на простите числа.

Ограничения

$2 \leq n \leq 100$

Всяко от дадените числа във входа не надминава 200 000.

[4] Пролетен турнир по информатика, Пловдив, 14 юни 2008 г.

Примери

Вход	**Вход**
4	10
1 36 9 1	1 2 3 2 1 3 1 13 28 4
Изход	**Изход**
0	5

Авторско решение

```
#include<iostream>
using namespace std;
int main()
{
int n,br=0;
int arr[100],flag;
int i,j;
cin>>n;
for(i=0;i<n;i++)
cin>>arr[i];
for(int j=0;j<n;j++)
{
 flag=1;
 for(int del=2;del<=arr[j]/2;del++)
   if(arr[j]%del==0)
   {
     flag=0;
     break;
   }
 if(flag && arr[j]>1)
 br++;
}
cout<<br<<"\n";
return 0;
}
```

Анализ на решението

Декларира се масив от 100 елемента (максималната стойност на `n`). С оператор за цикъл числата се записват в масива. За всяко от въведените числа, като се използва условен оператор и операция деление с остатък (%), се проверява дали числото се дели без остатък на числата от 2 до самото число, целочислено разделено на 2. Ако е така, то на променливата `flag` се присвоява стойност 0, което означава, че числото не е просто и се излиза от цикъла с оператор `break`.

Преди проверката на всяко от въведените числа на променливата `flag` се

присвоява стойност 1 и ако тя не се промени в тялото на цикъла, това показва, че числото е просто. Прави се проверка дали числото е по-голямо от 1, тъй като 1 не се приема за просто число. Двете условия се проверяват чрез условен оператор и оператор за логическо И (&&). Ако и двете условия са изпълнени, то броячът на простите числа `br` се увеличава с единица.

Задача 3. ОБИКОЛКА[5]

Приятелите на Лъчо му подарили за рождения ден котка, куче, морско свинче и костенурка. Лъчо се привързал много към тях и всеки ден ги разхождал в парка. Алеята за разходки представлявала затворена крива линия. Костенурката правела една обиколка за `x` минути, морското свинче – за `y` минути, котката – за `z` минути, а кучето – за `t` минути. Един ден Лъчо решил да пресметне след колко най-малко минути домашните му любимци ще се съберат отново на старта, ако тръгнат едновременно от едно и също място и в една и съща посока.

Вие можете да помогнете на Лъчо, като напишете програма *lap*, която по дадени естествени числа – `x`, `y`, `z` и `t` – (всяко от които се намира в интервала от 1 до 10000) намира и отпечатва търсения брой минути.

Вход

От първия ред на стандартния вход се въвеждат стойностите на числата x, y, z и t, разделени с по един интервал.

Изход

На един ред на стандартния изход програмата трябва да изведе едно число, равно на търсения брой минути.

Пример

Вход

10 12 15 20

Изход

60

Авторско решение

```
#include <iostream>
using namespace std;
int main()
{
long long i,k,a[4],j,d,d1,r,p;
for(i=0;i<4;i++)
cin>>a[i];
d1=a[0];
for(i=1;i<4;i++)
```

[5] Пролетен турнир по информатика, Пловдив, 14 юни 2008 г.

```
{
  j=d1;
  p=a[i];
  while(p!=0)
  {
      r=j%p;
      j=p;
      p=r;
  }
  d=j;
  k=(d1*a[i])/d;
  d1=k;
}
cout<<k<<endl;
return 0;
}
```

Анализ на решението

С оператор за цикъл в предварително деклариран масив от 4 елемента (в съответствие с броя на животните) се въвеждат времената на 4-те животни. Първата стойност от масива се присвоява на променливата `d1`. Организира се нов цикъл, започващ от втория елемент на масива. В тялото на този цикъл се организира вложен цикъл, в който се реализира търсенето на най-малкото число, което е кратно и на 4-те въведени времена.

Задача 4. ЦИФРИ[6]

Пешо Хакера си открил следната игра: намислял си две цели числа – `a` и `b` – и две цифри – `p1` и `p2`. След това записвал всички цели числа от интервала `[a;b]` и задрасквал цифрите им, които се делят на `p1` или на `p2`. После преброявал колко цифри са останали незадраскани.

Например при a = 15, b = 23, p1 = 2 и p2 = 3, в интервала [15;23] има общо 8 цифри, които не се делят на 2 или на 3: 15 ~~16~~ 17 ~~18~~ 19 ~~20~~ ~~21~~ ~~22~~ ~~23~~.

Напишете програма *digit*, която помага на г-н Хакера за решаване на тази задача.

Вход

На първия ред на стандартния вход се въвеждат `a`, `b`, `p1` и `p2`, разделени с интервал.

Изход

Програмата извежда на един ред на стандартния изход едно цяло число –

[6] VIII Национален есенен турнир по информатика и информационни технологии, „Джон Атанасов“, Шумен, 29.11.2008 г.

броя на останалите незадраскани цифри.

Ограничения:

$1 \le a \le b \le 1000$

$p1 > 1$

$p2 > 1$

Пример

Вход	**Изход**
94 106 4 5	20

Авторско решение

```
#include <iostream>
using namespace std;
int main()
{
   int a,b,p1,p2,br=0;
   int i,x,c;
   cin>>a>>b>>p1>>p2;
   for(i=a;i<=b;i++)
   {
     x=i;
     while(x!=0)
     {
       c=x%10;
       if(c%p1!=0 && c%p2!=0)
         br++;
       x=x/10;
     }
   }
   cout<<br<<endl;
   return 0;
}
```

Анализ на решението

С оператор за цикъл на променливата `x` се присвояват числата в интервала от `a` до `b`. В тялото на вложения цикъл посредством деление с остатък на 10 на променливата с се присвоява последната цифри на поредното число. С помощта на условен оператор се проверява дали тази цифра се дели с остатък на `p1` и на `p2`. Ако е така, то стойността на променливата брояч `br` се увеличава с единица.

След това чрез целочислено деление на 10 се премахва последната цифра от числото и ако получената стойност е различна от 0, за следващата цифра се повтаря вече описаната процедура.

Задача 5. СНЕЖАНКА[7]

Пътека в планината минавала през a на брой полянки. Снежанка се разхождала по пътеката и набрала голям букет цветя. От първата полянка тя откъснала едно цвете, от втората – две, от третата – три. Когато стигнала последната полянка, откъснала от нея a на брой цветя. Снежанка разпределила поравно събраните цветя между седемте джуджета, така че всяко да получи букет с нечетен брой. Напишете програма princess, която пресмята по колко цветя трябва да получи всяко джудже и колко цветя ще останат за Снежанка, така че за нея да са възможно най-малко.

Вход

На първия ред на стандартния вход се въвежда едно цяло число a – броят на полянките.

Изход

Програмата извежда на стандартния изход две цели числа, разделени с един интервал. Първото показва по колко цветя ще получи всяко от джуджетата, а второто – колко цветя са останали за Снежанка.

Ограничения

5<=a<=15

Примери

Вход	**Вход**
5	7
Изход	**Изход**
1 8	3 7

Авторско решение

```
#include<iostream>
using namespace std;
int main()
{
  int a,i,s=0;
  cin>>a;
  for(i=1;i<=a;i++)
    s=s+i;
  int d=s/7;
  int p=s%7;
  if(d%2==0)
  {
    d--;
    p=p+7;
  }
```

[7] XV Национална олимпиада по информатика, Общински кръг, 25.01.2009 г.

```
  cout<<d<<" "<<p<<endl;
}
```

Анализ на решението

С оператор за цикъл в променливата s се намира сумата на цветята, които е набрала Снежанка от всички полянки.

Пресмята се по колко цветя се падат на едно джудже. Резултатът се записва в променлива `d`. В случая делението е целочислена операция, защото операндите са цели числа. Когато се намери броят на цветята в букетчетата, се проверява дали е нечетен. Ако не е така, от всяко джудже се взема по едно цвете, като се намалява стойността на `d` с единица, а към `p` – броят на останалите цветя за Снежанка, се добавя 7.

1.3. Задачи за самостоятелна подготовка

Допълнителни задачи за самоподготовка могат да се намерят на сайта за национални състезания по информатика за ученици (http://www.math.bas.bg/infos/comp.html).

Допълнителна литература за подготовка

1. Василев, А. C++ в примери и задачи. Велико Търново: Асеневци, 2015.

2. Кисимов, В. Основи на алгоритмизацията и програмирането на основата на език JAVA. София: Изд. комплекс – УНСС, 2013.

3. Наков, Св. Въведение в програмирането с Java. София, 2008.

4. Сълов, В. Въведение в програмирането. Варна: Наука и икономика, 2015.

5. Тодорова, М. Структури от данни и програмиране на C++. София: Сиела, 2011.