Софийски университет „Св. Климент Охридски“
Факултет по математика и информатика

ЗАДАЧИ ЗА ДОМАШНО 3

курс Увод в програмирането
за специалности Информатика, Компютърни науки и Софтуерно инженерство
зимен семестър 2016/2017 г.

ОБЩИ ЗАБЕЛЕЖКИ

Съблюдавайте общите изисквания публикувани в Moodle.

Четете внимателно условията на задачите и проверявайте решенията си с примерните тестови данни. Ако смятате, че има грешка в тях, моля пишете на преподавателите. Всякакви въпроси по условията също отправяйте към тях.

При тестването на решенията ще се очаква те да завършват за по-малко от 15 секунди на всеки тест за всяка от задачите. Изведените от програмите ви данни трябва да се побират в 2 MB.

Всичко, което използвате, но не е преподавано на лекции и упражнения ще трябва да обясните на защитите. Например ако използвате функции от стандартната библиотека, ще се очаква, че можете да обясните как работят те, как сте стигнали до решението да ги използвате във вашия код и т.н.

Задача 1. (+1 Recursion)
Професор Х се забавлява като съставя интересни числови редици. Последното му творчество е следната редица:

1, 121, 1213121, 121312141213121, …

Първият член на редицата е 1. Всеки от следващите членове се получава от две копия на предходния член на редицата, между които е изписан поредният номер на текущия член. Тъй като дължината на числата в редицата нараства много бързо и ръчното им изписване е доста трудоемко, професорът се нуждае от програма, която да извежда посочения от него член на тази редица. Помогнете му като напишете програма, която по въведено цяло положително число **n** < 20 извежда **n**-тия член на редицата, следван от знак за нов ред.

Пример:

Вход	**Изход**
5	12131214121312151213121412131213121

Задача 2. (+1 Recursion, +1 Storing stuff in arrays)
Да се напише програма, която по въведено естествено число **number** ≤ 32 намира всички възможни "разбивания" на **number** като сума от естествени числа и извежда само уникалните такива представяния, като числата от всяко представяне са подредени в намаляваща редица, т.е. за всяко разбиване **{number = x_1 + x_2 + x_3 + … + x_n}** е в сила **x_1 ≥ x_2 ≥ x_3 ≥ … ≥ x_n**. Всяко разбиване трябва да бъде изведено във формата **{number = x_1 + x_2 + x_3 + … + x_n}** без къдравите скоби, където **x_1, …, x_n** са елементите на поредното разбиване на **number**. Подредете разбиванията лексикографски спрямо елементите им, започвайки от най-голямото, както е показано в примера по-долу. След всяко разбиване, включително и последното, да има знак за нов ред.

Пример:

Вход	**Изход**
5	5 = 5 5 = 4 + 1 5 = 3 + 2 5 = 3 + 1 + 1 5 = 2 + 2 + 1 5 = 2 + 1 + 1 + 1 5 = 1 + 1 + 1 + 1 + 1

Задача 3. (+1 Working with text)
Пешо много обичал да си прави каламбури с думите и като говори никой да не го разбира. Например, често взимал една дума и обръщал буквите ѝ. В други ситуации заменял едни букви с други. Обаче с времето започнал да се объркva и да му става все по-трудно да прави подобни каламбури без грешка. Сетил се, обаче, че с модерните в последно време компютри това може да стане много лесно. И тук вие, като негови приятели, ще се притечете на помощ. Пешо ви моли да направите програма, която да генерира подобни каламбури за него. Тази програма ще прочита от стандартния вход два реда. На първия ред ще е записан текста, който Пешо иска да промени (не по-дълъг от 10000 символа), а на втория ред ще е зададено едно естествено число **N**. "Завъртане" на дума наричаме преместване на последната ѝ буква преди първата. Вашата задача ще е да завъртите буквите на всяка дума от текста циклично N пъти и да изведете резултата. Например думата "Пешо" завъртяна един път изглежда "оПеш", 2 пъти: "шоПе", 3 пъти: "ешоП", 4 пъти: "Пешо", 5 пъти: "оПеш" и т.н. Дума, от своя страна, е максимална последователност от малки или главни латински букви, т.е. такава последователност, пред и след която НЯМА малка или главна латинска буква. Буквите в съобщението ще са латински, но се допуска използване на тире като буква, само в случай, че е между две латински букви.
Така ако Пешо иска да каже: *"Oh, jingle bells, jingle bells, jingle all the way. Oh, what fun it is to ride in a one horse open sleigh."* и N = 3 вие трябва да му подскажете:
"hO, glejin llsbe, glejin llsbe, glejin all the way. hO, hatw fun ti si ot ider ni a one rseho peno ighsle."

Пример:

Вход	**Изход**

`Pesho` `3`	`shoPe`
`Ana-Maria` `4`	`ariaAna-M`

Задача 4. (+1 Storing stuff in arrays)
Отново шах! В четвърта задача от Домашно 1 се проверяваше дали дадена фигура поставя царя под шах. Това е добра позиция в една партия, но изобщо не е задължително да е печеливша. Е, фигурата от Домашно 1, този път, е извикала подкрепление, за да матира царя. Затова трябва да се направи нова проверка, този път за шах, мат или пат на царя. Да се напише програма, която въвежда цяло число **n** (2 ≤ **n** ≤ 16), задаващо броя на атакуващите фигури, след което на **n** реда въвежда вида и координатите на всяка от тях. На последния ред се въвеждат координатите на атакувания цар.

Координатите на шахматната дъска са във вида - “X Y”, където:

- X е малка латинска буква от ‘a’ до ‘h’, означаваща поредна колона на шахматната дъска, отляво надясно
- Y е цифра от 1 до 8, означаваща пореден ред на шахматната дъска, отдолу нагоре

(Удобно описание на шахматната нотация можете да намерите тук.)

Атакуваща фигура може да бъде: дама, офицер, кон, топ или цар. Символите, съответстващи на валидните фигури са:

- ‘Q’ — дама (Queen)
- ‘B’ — офицер (Bishop)
- ‘N’ — кон (kNight)
- ‘R’ — топ (Rook)
- ‘K’ — цар (King)

Програмата да извежда на стандартния изход единствен ред, завършващ със знак за нов ред, на който е изписан един от следните текстове:

1. `"Checkmate!"`, ако царят е матиран.
2. `"Check!"`, ако царят е само в шах, но не в мат.
3. `"Stalemate!"`, ако царят е в патова ситуация, т.е. не е в шах, но няма съседна позиция, на която да може да се премести.
4. `"The show must go on!"`, ако царят не е в нито една от горните ситуации.

Да се отчете факта, че атакуващите фигури могат да си пречат една на друга. Ако например, между царя и атакуваща го дама има атакуващ го кон, то дамата не го застрашава.

Приемете, че всички фигури, ще бъдат коректно разположени на шахматното поле:

1. Всички въведени позиции ще бъдат валидни и уникални.
2. Всички въведени типове на фигурите ще бъдат валидни.
3. **Винаги** ще има **точно 1** цар на страната на атакуващите фигури.
4. Двата царя няма да са един до друг на шахматната дъска.

Примери:

Вход	Изход	Дъска
4 K e 8 B h 5 N e 4 R a 1 e 1	Checkmate!	Click
2 Q c 2 K b 3 c 1	Checkmate!	Click
4 N c 2 Q a 2 R h 1 K e 8 f 1	Check!	Click
2 K f 6 B f 7 f 8	Stalemate!	Click
5 Q a 8 K d 5 N g 2 N h 2 B g 1 h 1	The show must go on!	Click

Задача 5. (+1 Working with text)

Напишете програма, която оценява аритметични изрази, в които участват само неотрицателни цели числа и операцията събиране. Изразите изглеждат ето така:

<израз> = <число> + <число>{ + <число>}

Числата могат да бъдат записани в двоична, осмична, десетична или шестнайсетична бройна система. Числата, записани в двоична бройна система, имат префикс 0b, тези в осмична имат префикс само 0, а тези в шестнайсетична — 0x. Например, числото 29, записано в двоична система ще бъде “0b11101”, в осмична — “035”, в десетична — “29”, а в шестнайсетична — “0x1D”.

Програмата да въвежда от стандартния вход коректно записан аритметичен израз на един ред и да извежда на стандартния изход стойността на израза в десетична бройна система.

Ограничения:

1. Стойността на израза ще бъде число без знак, чиито двоичен запис се побира в 64 бита.
2. Изразът ще бъде съставен **само** от цифри, главните латински букви от `'A'` до `'F'` (вкл.), малките латински букви `'b'` и `'x'`, интервали и символа `'+'`.
3. Всички числа в израза ще бъдат коректно изписани в съответната бройна система.
4. Ако в записа на число има водещи нули, те да се игнорират. Обърнете внимание, че числата записани в десетична бройна система не могат да имат водещи нули, защото ще бъдат неразличими от тези, записани в осмична бройна система.
5. Буквите `'b'` и `'x'` ще бъдат използвани **само** в записа на съответно двоични и шестнайсетични числа, а главните букви от ‘A’ до ‘F’ — **само** в записа на шестнайсетични числа.
6. След всяко число, **освен последното**, и след всеки знак `'+'` ще има **точно един** интервал. Интервали на други места в израза не са допустими.
7. Изразът ще завършва с нов ред и няма да е по-дълъг от 1000 **символа**.

Примери:

Вход	**Изход**
`1 + 2 + 3 + 4 + 5 + 6 + 7 + 8 + 9 + 10`	`55`
`0b111111111111 + 20`	`4115`
`0b1011 + 0xF + 10 + 076`	`98`
`95647584 + 0b0 + 0x0 + 0`	`95647584`
`0x00000F + 004 + 120`	`139`

Задача 6. (+1 Memory Summoning, +1 Storing stuff in arrays, +1 Knowing Basic Algorithms)
Иванчо е асистент по програмиране на студенти от първи курс и едно от задълженията му е да проверява домашните на студентите. Докато е проверявал домашните, той си е записвал всяка оценка като ред във файл. Понеже редът, в който е проверявал домашните, е случаен, записите му са разбъркани. Накрая той трябва да нанася точките в Moodle, но понеже файлът не е подреден, това ще е трудно за него. Така Иванчо решава, че студентите ще подредят оценките си вместо него и пуска такава задача на следващото домашно. Помогнете на Иванчо да подреди оценките, така че да са най-удобни за нанасяне.

На стандартния вход се подава брой на редове с оценки във файла (неотрицателно число), а след това се въвежда всеки един ред с оценки.
Всеки ред (освен първия) във файла на Иванчо е от вида "fn n p", като:

- “fn" е факултетен номер (положително цяло число ≤ 1000000)
- "n" е номер на задача (положително цяло число, не повече от 10)

- "p" е брой точки за съответната задача (неотрицателно дробно число не повече от 2 с една цифра след десетичната запетая)

На стандартния изход програмата трябва да извежда всички редове с оценки, които са подредени както следва:

- всички редове да са подредени в нарастващ ред по факултетен номер
- всички редове за един и същ факултетен номер трябва да са подредени в нарастващ ред по номер на задача
- всеки факултетен номер има най-много един резултат за една задача

Примери:

Вход	**Изход**
6 45012 1 1.9 81233 5 1.5 62345 3 1.2 45014 3 0.9 81234 1 0.3 45013 9 1.6	45012 1 1.9 45013 9 1.6 45014 3 0.9 62345 3 1.2 81233 5 1.5 81234 1 0.3
4 45999 3 0.7 82233 4 1 45999 10 1.4 45999 5 1.8	45999 3 0.7 45999 5 1.8 45999 10 1.4 82233 4 1
8 814242 5 0 45369 3 2 1 1 1 65432 1 0 1234 5 1.5 1 2 0.3 45369 1 1.9 814242 3 1.7	1 1 1 1 2 0.3 1234 5 1.5 45369 1 1.9 45369 3 2 65432 1 0 814242 3 1.7 814242 5 0

Задача 7. (+1 Working with text)

Напишете преводач за Pig Latin. Това е “диалект” на английския език, който се получава като се разместят сричките в думите по следния начин:

1. Ако думата започва със съгласна буква, всички букви преди първата гласна се преместват в края на думата. След това се добавя сричката “ay” в края на думата. Например, думата “**banana**” ще стане “**ananabay**”, а думата “**trash**” - “**ashtray**”
2. Ако думата започва с гласна буква, просто се добавя сричката “way” в края на думата. Например “**omelet**”, ще стане “**omeletway**”

От програмата ви се очаква да въведе от стандартния вход правилно написан английски текст, не по-дълъг от 10000 символа на един ред и да изведе съответния му превод на Pig Latin също на един ред, завършващ със знак за нов ред. Допустими символи за входния текст са само малки и големи латински букви, цифри, интервали, кавички и препинателни знаци (**" ; . - , ! ?**). Програмата трябва да обработи **само** думите в текста. Интервалите, цифрите, кавичките и препинателните знаци трябва да се запазят без промяна.

Думите в текста ще бъдат или изцяло от малки букви, или изцяло от главни букви, или само с една главна буква в началото на думата. Ако дадена дума започва с главна буква, нейния превод също трябва да започва с главна буква. Например “**Pig**” се превежда “**Igpay**”. Преводът на думите, съставени изцяло от малки букви, трябва да бъде само с малки букви, а този на думите, съставени изяло от главни букви — само с главни букви.

Забележка 1: Към думите, съставени изцяло от главни букви, **НЕ** се включват еднобуквени думи. Т.е. Превода на “**I**” трябва да бъде “**Iway**”.
Забележка 2: Приемете, че ‘y’ винаги е гласна буква.

Примери:

Вход
`Pig Latin is a little bit strange.`
Изход
`Igpay Atinlay isway away ittlelay itbay angestray.`

Вход
`I already hate that "language"!`
Изход
`Iway alreadyway atehay atthay "anguagelay"!`

Вход
`“Pig Latin is a language game in which words in English are altered. The objective is to conceal the words from others not familiar with the rules.” - WIKIPEDIA`
Изход
`"Igpay Atinlay isway away anguagelay amegay inway ichwhay ordsway inway Englishway areway alteredway. Ethay objectiveway isway otay oncealcay ethay ordsway omfray othersway otnay amiliarfay ithway ethay ulesray." - IKIPEDIAWAY`

Задача 8. (+1 Storing stuff in arrays, +1 Knowing basic algorithms)

Да се напише програма, която прочита от стандартния вход цяло число **N**, 0 < **N** < 100000 и след това **N** цели числа. Програмата да извежда на стандартния изход въведените числа, подредени според остатъка си по модул 3 — първо тези, които се делят на три, после тези, които дават остатък 1 и накрая останалите. Ако две числа дават еднакъв остатък при деление на три, то те трябва да запазят относителния си ред при извеждането.

Примери:

Вход	**Изход**
8 1 2 3 4 5 6 7 8	3 6 1 4 7 2 5 8
3 2 1 3	3 1 2
6 12 33 11 8 0 -1	12 33 0 11 8 -1

Задача 9. (+1 Memory summoning, +1 Storing stuff in arrays, +1 Knowing basic algorithms)
В края на 2016 година, точно преди речта на президента, известната хакерка от Горно Побитово, Баба Пенка кротко дефейсвала сайта на Министерството на образованието. Тя правела това за пети пореден път за 2016 година, но трябвало да убие с нещо времето, преди да излезе навън и да се включи в голямото гърмене с напазарувания от нея арсенал, който превъзхождал отбраната на Лихтенщайн и Монако, взети заедно. В този момент тя се почувствала някак странно. Обзел я хлад, а необяснимо притеснение свило стомаха ѝ. Причината за това можела да е само една — Баба Пенка разбрала, че изтича договорът ѝ за мобилен телефон. Това означавало, че ще се наложи да отиде и да подпише нов договор с мобилния си оператор. Както е характерно за България, най-големите страхове на Баба Пенка били да не настине или да не се мине, а второто винаги ставало, когато отивала да подпише нов договор за мобилна услуга. Учтивите, но коварни служители в офисите на мобилния оператор винаги успявали да я подлъжат да подпише за повече, отколкото има нужда. Настиването също не можело да се изключи като възможност и настроението на Баба Пенка започнало да се влошава. Под пръстите ѝ сайтът на министерството придобивал все по-апокалиптичен вид. Тя обаче бързо се овладяла и решила, че на това трябва да се сложи край. Бързо нахвърляла спецификация за програма, която да избере вместо нея най-подходящ (тоест възможно най-евтин) план. Баба Пенка не иска да пише програмата сама, защото има много по-важни задачи в момента. Вашата задача е да напишете въпросната програма.

Програмата ви трябва да въвежда от стандартния вход цяло положително число **N**, означаващо броя на плановете, които мобилният оператор предлага (**N** < 100 000). След него се въвеждат данните за **N**-те плана, като за всеки план се въвеждат цена, брой включени минути, брой включени мегабайти и брой SMS-и. Накрая се въвежда потреблението, което Баба Пенка има. Планът, който ще изберете за нея, трябва да включва поне толкова на брой минути, мегабайти и SMS-и, колкото е нейното потребление. Данните за плановете и потреблението се описват с цели, неотрицателни числа, както следва:

- Цената може да бъде от 0 до 1 000 лева;
- Минутите, мегабайтите и SMS-ите могат да бъдат между 0 и 100 000.

При въвеждането на данните за плановете, не е задължително те да са в някакъв определен ред. Например възможно е най-напред да се въведе план за 30 лева, после за 10, после за 20 и т.н..

Не е задължително по-скъп план да превъзхожда по-евтин по всички характеристики. Например възможно е план за 20 лева да съдържа 500 минути и 100 мегабайта, а план за 25 лева да съдържа 350 минути и 350 мегабайта.

Програмата трябва да изведе на екрана данните на всички планове, които удовлетворяват търсеното потребление, подредени в нарастващ ред по цена (макар и пестелива, Баба Пенка иска да е добре информирана и да знае какво друго се предлага, за да не изпусне някоя голяма далавера). Ако има два плана с еднаква цена, изведете ги в реда в който са подадени на входа. Данните да се извеждат в същия ред, в който са въведени, разделени с интервали. Ако няма план, който удовлетворява изискванията, на стандартния изход трябва да се изведе текст `"No solution"`, следван от знак за нов ред.

Примери:

Вход	**Изход**
5 5 50 500 10 30 1000 3000 200 10 300 50 0 20 200 200 200 40 800 10000 500 250 100 10	30 1000 3000 200 40 800 10000 500
2 10 300 50 0 20 200 200 200 300 100 10	No solution

Задача 10. (+1 Storing stuff in arrays, +1 Working with text, +1 Knowing basic algorithms)
Нашият лирически герой Пешо обичал да си води дневник, но не искал никой да може да го прочете. Първо решил, че ще пише на английски, но се оказало, че го знаели прекалено много хора, затова измислил схема с която да кодира записките си. Преди да запише откровение за деня, той правил таблица 5 x 5 с латински букви, например:

A	B	C	D	E
F	G	H	I	J

K	L	M	N	O
P	Q	R	S	T
U	V	W	X	Y

Тази таблица оказвала как да кодира (съответно да декодира текста си). Имало няколко прости правила:

- всички букви от таблицата трябва да бъдат различни главни латински букви, което означава, че една от буквите остава незаписана в таблицата (в този случай **Z**)
- за да кодира текста, който той ще запише той го преработвал по следния начин:
    1. Определял някоя буква за **“Заместител”**, обикновено това е **X**, освен ако тя не отсъства от таблицата в такъв случай избирал **Z**.
    2. Ако текста е с нечетна дължина той добавял Заместителя в края на текста.
    3. Всички малки букви се заменят със съответните им главни.
    4. Всяко срещане на символ в текста, който не е от таблицата бива заменен със Заместителя.
- след това кодирането протича по следния начин:
    1. Текстът се разбива на последователни двойки букви.
    2. Всяка двойка се кодира в нова двойка букви:
        a. ако първоначалната двойка букви са в един и същи ред, то те биват заменени със буквите **непосредствено вдясно** от тях например, ако имаме горната таблица и двойката **IG** тя бива заменена от **JH**. Ако някоя от буквите е последна в реда, то тя се заменя с първата от същия ред например, ако имаме **NO**, то те се заменят със **OK**.
        b. ако буквите са в една и съща колона правим същото като в предния пример, но сменяме буквите със тези които са **непосредствено под** тях
        c. ако буквите са в различен ред и колона, то местата им образуват правоъгълник. В този случай заменяме първоначалната двойка с буквите от другите два ъгъла на правоъгълника, като всяка от първоначалните букви заменяме със срещуположния ъгъл на правоъгълника на същия ред. Например **GS** бихме заменили със **IQ**, а **IC** с **HD**.
        d. ако двойката се състои от еднакви букви които са различни от Заместителя то втората бива заменена със него и кодирането на двойката продължава по стандартния начин. Например **EE**, първо става **EX**, а после това бива кодирано до **DY**.
        e. ако двойката се състои от еднакви букви които са Заместители, то те не се кодират, а остават в същия вид

Е, процедурата за кодиране не била никак лесна и скоро на Петър му омръзнало да я прави на ръка и затова ви поръчал да направите програма, която да го прави вместо него. Това което трябва да прави програмата е въвежда от стандартния вход 5 реда по 5 символа, разделени с интервали (таблицата за кодиране), а на шестия ред въвежда текста, който трябва да кодирате. Програмата трябва да изведе на стандартния изход кодирания текст по съответните правила. Ако въведената таблица не се състои от 25 различни главни латински букви, да се изведе текста `"Bad table!"`, следван от знак за нов ред.

Примери:

Вход	**Изход**
A B C D E F G H I J K L M N O P Q R S T U V W X Y Bad food	CBIDJKNE
Z Y X W V U T S R Q P O N M L K J I H G F E D C B fo foo fooo	EPZDNYZDNYNY
X X A X X X A M A X A X M X A X A M A X A X T X A It is hard to draw with letters	Bad table!