POLITECHNIKA ŚLĄSKA

Wydział Matematyki Stosowanej
Studenckie Koło Naukowo-Informatyczne
„Link”
ul. Kaszubska 23, 44-100 Gliwice

ZADANIE 1 – SZYFR

Zadanie zaproponował: dr inż. Mariusz Pleszczyński, Wydział Matematyki Stosowanej, Politechnika Śląska

Jednym z prostszych szyfrów jest szyfr polegający na permutacji liter, gdzie każdej literze odpowiada pewna (często inna) litera. Przykładowo, jeśli każdej z liter przypiszemy kolejną literę alfabetu, to słowo *algorytmion* zaszyfruje się do słowa *ąłhószunjóń*. Taka permutacja, przy założeniu, że każdej literze przyporządkujemy odpowiadający jej numer występowania w alfabecie (a dodatkowo spacji przypiszemy liczbę zero):

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
	a	ą	b	c	ć	d	e	ę	f	g	h	i	j	k	l	ł
17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	
m	n	ń	o	ó	p	r	s	ś	t	u	w	y	z	ź	ż	

mogłaby być opisana równaniem $f(n) = (n+1)_{\mathrm{mod}\ 33}$, gdzie n jest numerem dowolnej polskiej litery, $a_{\mathrm{mod}\ b}$ oznacza resztę z dzielenia liczby a przez b, a wartość funkcji f określa, jaką literą zastąpimy literę o numerze n (poprzez znalezienie jej numeru).

Możemy przyjąć ogólniej, że permutacja polskiego alfabetu odbywać się będzie poprzez funkcję

$$f(n) = (p \cdot n + q)_{\mathrm{mod}\ 33}$$

gdzie $p, q \in \mathbb{N}$, $n \in \{0,1,2,\dots,32\}$. Przykładowo, funkcja $f(n) = (17n+50)_{\mathrm{mod}\ 33}$ literę *ę* (o numerze 8) zaszyfruje literą *ó* (o numerze 21), bo

$$(17 \cdot 8 + 50)_{\mathrm{mod}\ 33} = (186)_{\mathrm{mod}\ 33} = (5 \cdot 33 + 21)_{\mathrm{mod}\ 33} = 21.$$

Napisz program, który dla zadanego ciągu znaków odszyfruje ten ciąg znaków, przy założeniu, że zaszyfrowany on został funkcją $f(n) = (p \cdot n + q)_{\mathrm{mod}\ 33}$.

Zakładamy, że szyfrowany tekst składa się ze słów znajdujących się w ustalonym słowniku (patrz zadanie 3 – Kalambury).

Program od momentu podania ciągu znaków do podana odszyfrowanego tekstu nie może korzystać z pomocy człowieka.

Dla krótkich ciągów znaków odszyfrowywanie może być niejednoznaczne, ale wraz ze wzrostem długości zaszyfrowanego ciągu prawdopodobieństwo jednoznaczności jest praktycznie pewne.

Przykładowo, po podaniu ciągu znaków *śnanłerlmńnem łdthłtalmhńłbócbźzpź* program odszyfruje go jako *będę zwycięzcą tej edycji konkursu* (funkcja szyfrująca w tym przypadku ma postać $f(n) = (25n+16)_{\mathrm{mod}\ 33}$).

ZADANIE 2 – BILIARD

Zadanie zaproponował: dr inż. Mariusz Pleszczyński, Wydział Matematyki Stosowanej, Politechnika Śląska

Niech dany jest prostokąt o bokach długości 4 i 2, który uważać będziemy za powierzchnię stołu bilardowego. Na stole tym umieszczamy w dowolnym jego miejscu (poza brzegiem, tzn. w punkcie wewnętrznym tego prostokąta) kulę bilardową (bilę). Następnie bilę tę uderzamy z daną siłą w danym kierunku. Interesuje nas droga, jaką pokona uderzona bila.

Na potrzeby programu zakładamy, że środek stołu jest początkiem układu współrzędnych, że siła uderzenia bili (bilę traktujemy jako punkt) jest równoważna z drogą, jaką przebyłaby ta bila na stole o nieskończonej powierzchni (bez brzegów) – np. dla siły 20 bila przebyłaby 20 jednostek długości (5 razy dalej niż wynosi długość wyjściowego stołu). Zakładamy również, że kierunek uderzenia określany będzie poprzez wektor (kierunek wektora) o początku w punkcie położenia bili i o końcu w punkcie określonym podanymi dwoma współrzędnymi. Bila po dotarciu do brzegu stołu odbija się od niego (zgodnie z zasadami fizyki) i wytraca na tym odbiciu 5% swojej energii (tzn. gdyby bila uderzona byłaby z siłą 22, przebyłaby 2 jednostki długości natrafiając na brzeg, to po spotkaniu z tym brzegiem nie zostałoby jej 20 jednostek długości do przebycia, ale o 5% mniej, czyli 19 jednostek długości). Zakładamy też, że stół może być większy niż bazowy (który miał wymiary 4×2), a za nową wielkość stołu odpowiada argument (mnożnik) programu $m \geq 1$. Jeśli podalibyśmy wartość $m = 3$, to stół miałby wymiary 12×6, czyli byłby prostokątem

$$P = \{(x, y) \in \mathbb{R}^2 : -6 \leq x \leq 6, -3 \leq y \leq 3\}.$$

Podsumowując, zadaniem jest napisanie programu, który będzie miał sześć argumentów: $\mathbb{R} \ni m \geq 1$ – mnożnik odpowiadający za wielkość stołu; $\mathbb{R} \ni s > 0$ – siła uderzenia bili; x, y – współrzędne początkowego położenia bili na stole, gdzie $(x, y) \in P$ (dokładniej – należy do wnętrza P); $wx, wy \in \mathbb{R}$ – współrzędne wektora $[wx, wy]$ określającego początkowy kierunek poruszania się bili.

Program ma zwracać rysunek (przedstawiony w konsoli) drogi, jaką przebędzie bila po tak zdefiniowanym stole, poruszająca się z podanego punktu początkowego, uderzona z podaną siłą w kierunku (początkowym) podanego wektora.

ZADANIE 3 – KALAMBURY

Zadanie zaproponował: dr inż. Mariusz Pleszczyński, Wydział Matematyki Stosowanej, Politechnika Śląska

Jednym z zadań szaradziarskich jest kalambur. Jest to zagadka słowna, najczęściej dwuwiersz, w którego jednym wierszu podaje się znaczenie szukanego słowa, a w drugim znaczenia dwóch (lub większej liczby) słów składających się na rozwiązanie (szukane słowo). Przykładowo kalambur:

Posiada krowy przodka,
część uczniów w maju spotka.

ma rozwiązanie *matura* – posiada, to inaczej ma, a przodek krowy to tur (tu w dopełniaczu: krowy przodka, to tura), a to, co część uczniów spotka w maju, to matura.

Tak opisany kalambur jest kalamburem klasycznym. Znane są również jeszcze m.in. kalambury wiązane, w których słowa składające się na rozwiązanie „zazębiają” się ze sobą (słowo *matura* również jest dobrym przykładem – zazębiają się słowa *mat* oraz *tura* – oczywiście należałoby tu wymyślić inny dwuwiersz). Znane są również kalambury szufladkowe, gdzie jedno ze słów składających się na rozwiązanie „wsuwa” się do drugiego słowa. Tu również (po napisaniu odpowiedniego dwuwiersza) dobrym przykładem jest słowo *matura*. Do słowa *mara* wkładamy (po drugiej jego literze) słowo *tu*. Nas interesować będą w tym zadaniu właśnie kalambury szufladkowe.

W pliku *słownik.txt* znajduje się słownik, z którego korzystał będzie poniżej wspomniany program, w którym słowa (każde w nowej linii) posortowane są alfabetycznie.

Napisz program, który będzie sprawdzał, czy podany ciąg znaków (jedno słowo o długości co najmniej czterech znaków) jest kalamburem szufladkowym. Jeśli tak, to program poda dwa słowa, które składają się na podany ciąg znaków, który również musi być słowem znajdującym się w słowniku. Przykładowo dla argumentu *matura* program napisze, że podane słowo jest kalamburem szufladkowym i poda (przykładowo, bo rozkład taki nie musi być jednoznaczny) słowa *mara* i *tu.* Dla ciągu znaków *parabola* program napisze, że podane słowo jest kalamburem szufladkowym i poda słowa *pola* i *arab.* Dla argumentu *chrząszcz* program napisze, że nie jest to kalambur szufladkowy. Dla argumentu *kpiesot* dostaniemy taką samą odpowiedź (chociaż mamy tam słowo *pies* zanurzone w słowie *kot,* to ciąg znaków *kpiesot* nie jest słowem, bo nie ma go w słowniku).

Jeżeli program na wejściu dostanie ciąg długości dwa, to sam będzie w słowniku poszukiwał słów, które są kalamburami szufladkowymi o literze początkowej będącej pierwszym znakiem wejściowego ciągu i ostatniej literze będącej drugim jego znakiem. Przykładowo podając na wejściu *ra*, program mógłby zwrócić słowo (pewnie jest więcej takich słów) *rzekotka*, bo w słowie *rzeka* chowa się słowo *kot*. Dla argumentu *ąd* program napisze, że nie ma takiego kalamburu szufladkowego.

ZADANIE 4 – MONEY, MONEY, MONEY

Zadanie zaproponował: dr inż. Mariusz Pleszczyński, Wydział Matematyki Stosowanej, Politechnika Śląska

Mamy woreczek, w którym jest n monet o nominałach 1 zł, 2 zł i 5 zł składających się na pewną sumę k zł.

Napisz program, który po podaniu wartości n i k podawał będzie ile jest monet danego typu.

Ponieważ dla różnych argumentów wejściowych mogą być różne sytuacje: rozwiązanie nie istnieje, istnieje dokładnie jeden zestaw monet lub istnieje więcej niż jeden taki zestaw monet, to program ma odpowiednio zareagować na każdą sytuację – w pierwszym przypadku program napisze, że taka sytuacja jest niemożliwa, w drugim wypisze to dokładnie jedno rozwiązanie, a w trzecim wypisze wszystkie rozwiązania.

Przykładowo, podając $n = 7$ i $k = 33$, to program poinformuje nas o tym, że taka sytuacja jest niemożliwa; podając $n = 23$ i $k = 98$, to program poinformuje nas o tym, że w woreczku są 2 monety 1 zł, 3 monety 2 zł i 18 monet 5 zł; natomiast podając, że $n = 14$ i $k = 27$, to program poinformuje nas o tym, że w woreczku jest 1 moneta 1 zł, 13 monet 2 zł i 0 monet 5 zł lub 4 monety 1 zł, 9 monet 2 zł i 1 moneta 5 zł, lub 7 monet 1 zł, 5 monet 2 zł i 2 monety 5 zł, lub 10 monet 1 zł, 1 moneta 2 zł i 3 monety 5 zł.

ZADANIE 5 – AGENT SMITH

Zadanie zaproponował: dr inż. Mariusz Pleszczyński, Wydział Matematyki Stosowanej, Politechnika Śląska

Liczbą Smitha nazywany taką liczbę naturalną złożoną, której suma cyfr jest równa sumie cyfr wszystkich swoich dzielników pierwszych.

Przykładowo liczba 728 jest liczbą Smitha, bo $7 + 2 + 8 = 17$, dodatkowo wiemy, że $728 = 2^3 \cdot 7 \cdot 13 = 2 \cdot 2 \cdot 2 \cdot 7 \cdot 13$ oraz $2 + 2 + 2 + 7 + 1 + 3 = 17$.

Napisz program, który weryfikował będzie, czy podana liczba naturalna jest liczbą Smitha.

Przykładowo, podając liczbę 728, program napisze, że liczba ta jest liczbą Smitha, a podając jako argument liczę 777, program napiszę, że nie jest to liczba Smitha ($7 + 7 + 7 = 21$, a $777 = 3 \cdot 7 \cdot 37$ oraz $3 + 7 + 3 + 7 = 20 \neq 21$). Podobnie będzie dla liczby 787 (chociaż liczba ta ma sumę cyfr taką jak suma cyfr jej dzielników pierwszych, to nie uznajemy jej za liczbę Smitha, bo nie jest to liczba złożona).

Pisząc program staraj się nie korzystać z dodatkowych bibliotek matematycznych.