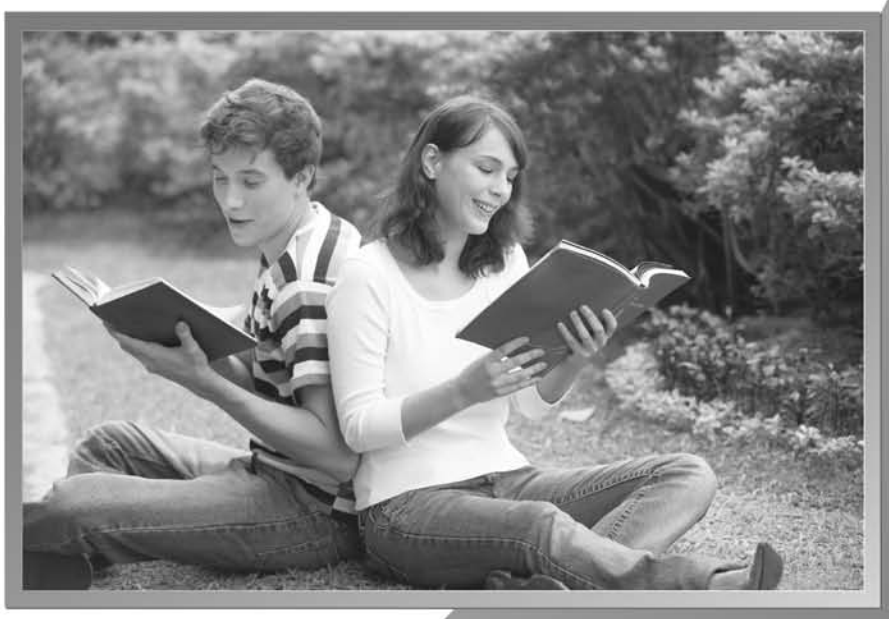
Osiągnięcia maturzystów w roku 2008

Komentarz do zadań z informatyki



Sprawozdanie z egzaminu maturalnego w maju 2008 roku

Opracowanie

Jan Chyży

Dorota Roman-Jurdzińska

Współpraca

Elżbieta Wierzbicka

Agata Kordas

Konsultacja naukowa

prof. dr hab. Krzysztof Diks

dr Janusz Jabłonowski

WSTĘP

Egzamin maturalny z informatyki odbył się w całym kraju w dniu 21 maja 2008 r. Maturzyści mogli go zdawać wyłącznie jako przedmiot dodatkowo wybrany, na poziomie rozszerzonym. Dla informatyki, jako przedmiotu zdawanego dodatkowo nie określono progu zaliczenia, ale wynik egzaminu został zapisany na świadectwie dojrzałości w skali procentowej.

OPIS ARKUSZY EGZAMINACYJNYCH

Arkusze egzaminacyjne zostały przygotowane zgodnie z koncepcją przedstawioną w *Informatorze o egzaminie maturalnym w 2008 roku – informatyka*.
Egzamin trwał 240 minut i składał się z dwóch części:

- część pierwsza egzaminu (pisemna) trwała 90 minut i polegała na rozwiązaniu bez użycia komputera arkusza zawierającego 3 zadania; za ich rozwiązanie można było uzyskać maksymalnie 40% ogólnej liczby punktów z całego egzaminu (40 punktów),
- część druga egzaminu (praktyczna) trwała 150 minut i polegała na rozwiązaniu przy użyciu komputera arkusza zawierającego 3 zadania; za ich rozwiązanie można było uzyskać maksymalnie 60% ogólnej liczby punktów z całego egzaminu (60 punktów).

W czasie trwania drugiej części egzaminu zdający pracowali przy autonomicznych stanowiskach komputerowych i mogli korzystać z danych zapisanych na płycie CD podpisanej *DANE*, która była dołączona do każdego arkusza. Każde stanowisko zdającego wyposażone było w oprogramowanie, które zdający wybrał z listy ogłoszonej przez dyrektora Centralnej Komisji Egzaminacyjnej na stronie internetowej CKE. Lista ta przedstawiała się następująco:

Środowisko	Język programowania (kompilator)*	Program użytkowy*
Windows z systemem plików NTFS	– Turbo Pascal 5.5 lub nowszy – Free Pascal (FPC 2.0) lub nowszy – MS Visual Studio .NET C++ – Borland C++ Builder 6 Personal – Dev C++ 4.9.9.2 lub nowszy – Delphi 7 Personal – MS Visual Studio .NET VB	– MS Office 2000 lub nowszy (w tym: Word, Excel, Access, PowerPoint)
Linux z KDE	– FreePascal (FPC 2.0) lub nowszy – GCC 4.1.1 C/C++ lub nowszy	– OpenOffice i MySQL 5.0 lub nowszy

*tylko jeden dla wybranego środowiska

Nadmienić należy, że większość zdających zdecydowała się zdawać tę część egzaminu w środowisku Windows, kilka procent z ogólnej liczby zdających wybrało środowisko Linux.

Arkusze egzaminacyjne i przykładowe ich rozwiązania dostępne są na stronie internetowej www.cke.edu.pl.

Opis zadań egzaminacyjnych, sprawdzane umiejętności, typowe odpowiedzi i uwagi do rozwiązań maturzystów.

CZĘŚĆ I

Zadanie 1. Potęgi (*14 pkt*)

W poniższej tabelce podane są wartości kolejnych potęg liczby 2:

k	0	1	2	3	4	5	6	7	8	9	10
2^k	1	2	4	8	16	32	64	128	256	512	1024

Ciąg $a=(a_0, a_1, a_2,...)$ definiujemy następująco:

a_k = reszta z dzielenia liczby 2^k przez 10 dla $k = 0, 1, 2, ...$

a) Korzystając z definicji, podaj 16 pierwszych wyrazów ciągu *a*. Wyniki umieść w poniższej tabelce:

k	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
a_k																

Uwaga: w dalszej części tego zadania możesz przyjąć, że operacje arytmetyczne na liczbach całkowitych (dodawanie, odejmowanie, mnożenie, dzielenie całkowite, reszta z dzielenia) wykonywane są w czasie stałym, niezależnie od wielkości argumentów.

b) W wybranej przez siebie notacji (lista kroków, schemat blokowy lub język programowania) podaj algorytm, który dla danej nieujemnej liczby całkowitej k wyznacza resztę z dzielenia liczby 2^k przez 10. Np. dla $k = 15$ wynikiem działania Twojego algorytmu powinno być 8.

Przy ocenie Twojego rozwiązania będzie brana pod uwagę zarówno poprawność zaproponowanego algorytmu, jak i jego złożoność czasowa, czyli liczba operacji arytmetycznych wykonywanych w trakcie obliczania wyniku.

Specyfikacja:

Dane: Liczba całkowita $k \geq 0$.

Wynik: Reszta z dzielenia 2^k przez 10.

c) Podaj w wybranej przez siebie notacji (lista kroków, schemat blokowy lub język programowania) algorytm obliczania liczby a^n, gdy a jest liczbą całkowitą, natomiast n jest potęgą liczby 2 ($n = 2^k$ dla pewnej liczby całkowitej $k \geq 0$). Przy ocenie Twojego rozwiązania będzie brana pod uwagę złożoność czasowa (w zależności jedynie od n) zaproponowanego algorytmu, czyli liczba operacji arytmetycznych wykonywanych w trakcie obliczania wyniku.

Wskazówka: zauważ, że $a^n = a^{\frac{n}{2}} \cdot a^{\frac{n}{2}}$, dla $n>1$.

Specyfikacja:

Dane: Liczby całkowite *a* i *n*, gdzie $n = 2^k$ dla pewnej liczby całkowitej $k \geq 0$.

Wynik: Liczba a^n.

Sprawdzane umiejętności
W zadaniu były badane umiejętności z I i II obszaru standardów. Zdający:

- rozwiązuje zadanie poprzez skorzystanie ze znanej metody
- dokonuje analizy zadania, opracowuje algorytm zgodny ze specyfikacją zadania i zapisuje go w wybranej przez siebie notacji
- analizuje liczbę działań wykonywanych w algorytmie

Rozwiązywalność zadania
33%, przy czym
a) – 74%
b) – 37%
c) – 19%

Typowe poprawne odpowiedzi zdających
a)

k	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
a_k	*1*	*2*	*4*	*8*	*6*	*2*	*4*	*8*	*6*	*2*	*4*	*8*	*6*	*2*	*4*	*8*

b)

krok 1: jeżeli k = 0, to wynikiem jest 1
krok 2: w przeciwnym przypadku
krok 2.1: policz resztę z dzielenia k przez 4
krok 2.2: jeżeli reszta = 0, to wynikiem jest 6
krok 2.3: jeżeli reszta = 1, to wynikiem jest 2
krok 2.4: jeżeli reszta = 2, to wynikiem jest 4
krok 2.5: jeżeli reszta = 3, to wynikiem jest 8

c)

krok 1: p := a
krok 2: dopóki n > 1 wykonuj
*krok 2.1: p := p * p*
krok 2.2: n := n div 2
krok 3: wynikiem jest p

Najczęściej powtarzające się błędy
W punkcie a) zdarzały się rozwiązania świadczące o braku zrozumienia definicji ciągu, np. jako wyrazy ciągu wynikowego przedstawiano pary liczb w postaci (k, a^k).
W punkcie b), w wielu rozwiązaniach (uznawanych za poprawne), zdający nie zdawali sobie sprawy z tego, że złożoność obliczania 2^k nie jest stała (jako jeden krok algorytmu podawali: *podnieść 2 do potęgi k*). Część zdających nie znała operacji obliczania reszty z dzielenia lub nie rozumiała różnicy między DIV a MOD. Innym błędem było sprowadzanie zadania do innego problemu niż podany w zadaniu, np. obliczanie ilorazu *liczba*/10 dla *liczba*=2^k, a następnie podawanie, że wynik jest równy pierwszej cyfrze po przecinku uzyskanego wyniku (nie pokazując, jak wyznaczyć wartość pierwszej cyfry po przecinku).

Częstym błędem popełnianym przez zdających w punkcie c) było przyjmowanie przy obliczaniu potęgi, że parametr k jest znany (co jest niezgodne z podaną specyfikacją) i bezpośrednie odwoływanie się do wartości k. Zdarzały się też rozwiązania, w których zdający czytali n, a następnie sprawdzali, czy n jest potęgą dwójki. Ponieważ własność $n=2^k$ dla naturalnego k wynikała ze specyfikacji zadania, sprawdzanie takie nie było konieczne.

Komentarz
Zadanie okazało się trudne dla maturzystów, zwłaszcza punkt c), który był najtrudniejszy, wypadł najsłabiej. Zadanie zostało skonstruowane poprawnie, o czym może świadczyć fakt, że najłatwiejszy był punkt a), w którym zdający miał podać 16 pierwszych wyrazów ciągu a, gdzie a_k = reszta z dzielenia liczby 2^k przez 10 dla k = 0, 1, 2, ... W punkcie b) poprawne rozwiązanie o stałej złożoności, które wymagało jedynie zauważenia własności definiowanego ciągu (dla kolejnych k=1, 2, 3, …, n, ciąg składa się z powtarzających się liczb: 2, 4, 8, 6), występowało znacznie rzadziej niż algorytm obliczania wyrazu ciągu „z definicji". Egzaminatorzy byli zaskoczeni, że zdający nie zdają sobie sprawy ze złożoności obliczania potęgi 2 do k-tej i uznają to za operację elementarną. W punkcie c) tylko część zdających skorzystała ze wskazówki i zoptymalizowała liczbę operacji arytmetycznych, uzyskując logarytmiczną złożoność czasową.

Zadanie 2. Słowa (*14 pkt*)

Niech $A = \{a, b\}$ będzie dwuliterowym alfabetem. Napisem nad alfabetem A nazywamy skończony ciąg znaków z tego alfabetu o długości większej od zera. Np. takimi napisami są:

a, *ab*, *aba*, *baba*, *aaaa*

Długość napisu w będziemy oznaczać przez $|w|$. Zatem $|aba| = 3$.

Jeżeli w_1 i w_2 są napisami, to przez w_1w_2 będziemy oznaczali napis zbudowany z napisu w_1 i z następującego po nim napisu w_2. Np. dla $w_1 = ab$ i $w_2 = aa$, $w_1w_2 = abaa$.

Zdefiniujemy teraz napisy 2-regularne. Każdy napis złożony tylko z jednej litery jest 2-regularny. Jeżeli napis w jest 2-regularny, to napis ww jest też 2-regularny. Żadne inne napisy nie są 2-regularne.

Oto procedura rekurencyjna **2REG**(w), która sprawdza, czy dany napis w nad alfabetem A jest 2-regularny.

Specyfikacja:

Dane: napis w o długości n ($n \geq 1$), składający się z liter należących do alfabetu A.
Wynik: odpowiedź *TAK*, jeśli napis w jest napisem 2-regularnym; odpowiedź *NIE*, jeśli napis w nie jest napisem 2-regularnym.

2REG(w);
krok 1: jeśli $|w| = 1$, to wynikiem jest *TAK*
krok 2: jeśli $|w| > 1$ i $|w|$ jest nieparzyste, to wynikiem jest *NIE*
krok 3: jeśli $|w| > 1$ i $|w|$ jest parzyste, to:
krok 3.1: podziel napis w na dwa napisy w_1 i w_2 o takiej samej długości i takie, że $w = w_1w_2$
krok 3.2: jeśli $w_1 \neq w_2$, to wynikiem jest *NIE*
krok 3.3: wynikiem jest wynik wywołania **2REG**(w_1)

a) Wypisz parametry wszystkich wywołań rekurencyjnych funkcji **2REG** dla poniższych napisów oraz podaj wynik jej działania:
   i. *aabbaabb*
   ii. *aaaaaaaa*
   iii. *bbbbbbbbbbbbbbbbbbbb*

np.: dla napisu *w* = *abab*, parametry wszystkich wywołań rekurencyjnych funkcji **2REG** i wynik jej działania są następujące:

abab→*ab*→*NIE*

b) Jakiej długości są napisy 2-regularne? Odpowiedź uzasadnij.

c) Ile jest napisów 2-regularnych o długości *n* ($n \geq 1$) nad alfabetem *A*? Odpowiedź uzasadnij.

d) Pewnym uogólnieniem napisów 2-regularnych są napisy 3-regularne.
Każdy napis jednoliterowy jest 3-regularny. Jeśli napis *w* jest 3-regularny, to każdy z napisów *wxw*, *wwx*, gdzie *x* jest dowolnym napisem nad alfabetem *A* i takim, że długość *x* jest taka sama jak długość *w*, jest napisem 3-regularnym. Żaden inny napis nie jest 3-regularny.
Przykładowymi napisami 3-regularnymi są: *a*, *aba*, *abaabaaaa*.
Ale *aaaabaaba* nie jest 3-regularny.

Napisz w wybranej przez siebie notacji (lista kroków, schemat blokowy lub język programowania) algorytm zgodny ze specyfikacją, który sprawdza 3-regularność danego napisu.

Specyfikacja:
Dane: napis *w*, o długości *n* ($n \geq 1$), składający się z liter należących do alfabetu *A*.
Wynik: odpowiedź *TAK*, jeśli napis *w* jest napisem 3-regularnym; odpowiedź *NIE*, jeśli napis *w* nie jest napisem 3-regularnym.

Sprawdzane umiejętności
W zadaniu były badane umiejętności z I i II obszaru standardów. Zdający:
- zna algorytm rekurencyjny i różne sposoby jego zapisu
- wyodrębnia kolejne parametry wywołań algorytmu rekurencyjnego
- dokonuje analizy zadania
- opracowuje algorytm zgodny z podaną specyfikacją i zapisuje go w wybranej przez siebie notacji

Rozwiązywalność zadania
43%, przy czym
a) 68%
b) 46%
c) 23%
d) 36%

Typowe poprawne odpowiedzi zdających
a)
$aabbaabb \rightarrow aabb \rightarrow NIE$
$aaaaaaaa \rightarrow aaaa \rightarrow aa \rightarrow a \rightarrow TAK$
$bbbbbbbbbbbbbbbbbbbb \rightarrow bbbbbbbbbb \rightarrow bbbbb \rightarrow NIE$

b)
Długość napisu musi być potęgą liczby 2, gdyż napis jednoliterowy jest 2-regularny, a każdy napis 2-regularny o długości większej od 1 powstaje z połączenia dwóch napisów 2-regularnych o takiej samej długości, a zatem ma długość dwa razy większą od długości każdego z tych napisów.

c)
Są tylko dwa napisy 2-regularne o długości n, gdy n jest potęgą liczby 2. Jeden napis to napis składający się tylko z liter a, drugi napis to napis składający się tylko z liter b. Jeśli n nie jest potęgą liczby 2, to nie ma napisów 2-regularnych o tej długości.
Jednoliterowy napis 2-regularny składa się albo z litery a, albo z litery b. Każdy napis 2-regularny o długości większej od 1 powstaje z połączenia dwóch identycznych, zbudowanych z tej samej litery, napisów 2-regularnych.

d)
3REG(w)
krok 1: n := długość słowa w
krok 2: jeżeli n = 1, to wynikiem jest TAK
krok 3: jeżeli n > 1 i n nie jest podzielne przez 3, to wynikiem jest NIE
krok 4: jeżeli n jest podzielne przez 3, to:
krok 4.1: podziel słowo w na 3 podsłowa w1, w2, w3 o równych długościach i takie, że w = w1 w2 w3
krok 4.2: jeżeli (w1=w2) lub (w1=w3), to wynikiem jest wynik wywołania 3REG(w1)
krok 4.3: w przeciwnym razie wynikiem jest NIE

Najczęściej powtarzające się błędy
W punkcie a) zdarzało się, że zdający podawali tylko jedno wywołanie funkcji 2REG(w), pomijając wywołania 2REG(w_1), co może świadczyć o niezrozumieniu działania rekurencji.
W punkcie b) najczęstsze błędne odpowiedzi to *napisy o długości parzystej* lub *napisy nieskończenie długie*.
Bardzo często występującą błędną odpowiedzią wśród zdających do punktu c) była odpowiedź: *nieskończenie wiele* (mylenie liczby napisów z ich długością).
W punktach b) i c) uzasadnienia często były niejasne albo nie było ich w ogóle, zdarzały się również błędne uzasadnienia do dobrej odpowiedzi.
W punkcie d) zdający wzorowali się na funkcji 2REG(w) i stąd wynikały najbardziej typowe błędy: badanie parzystości (nieparzystości) długości napisu zamiast sprawdzenia, czy jego długość jest podzielna przez 3, bądź też niepoprawne formułowanie warunku określającego, czy napis w_1 równa się napisowi w_2 lub w_3 (nieprawidłowy operator logiczny, tzn. $w_1=w_2$ **i** $w_1=w_3$ albo zamiennie $w_1\neq w_2$ **lub** $w_1\neq w_3$, a czasami wprowadzanie dodatkowych warunków niezgodnych z treścią zadania, np. $w_1=w_2$ **i** $w_1=w_3$ **i** $w_2\neq w_3$).
Innym błędem pojawiającym się w rozwiązaniach tego punktu zadania był brak wywołania funkcji 3REG dla w_1.

Komentarz
Zadanie okazało się dość trudne, zwłaszcza punkt c). Zdający, którzy dokonali prawidłowej analizy podanego w treści zadania algorytmu, nie mieli problemu z wypisaniem parametrów wszystkich wywołań rekurencyjnych funkcji 2REG(w). Podawali też dobre odpowiedzi do podpunktów b) i c) wraz z poprawnym uzasadnieniem. Ponadto, wzorując się na procedurze rekurencyjnej 2REG(w) poprawnie zapisali algorytm sprawdzający 3-regularność danego napisu zgodnie z przedstawioną specyfikacją. Zadanie podobało się zarówno zdającym, jak i egzaminatorom sprawdzającym prace.

Zadanie 3. Test (*12 pkt*)

Podpunkty a) – l) zawierają po trzy odpowiedzi, z których każda jest albo prawdziwa, albo fałszywa. Zdecyduj, które z podanych odpowiedzi są prawdziwe (**P**), a które fałszywe (**F**). **Zaznacz znakiem X** odpowiednią rubrykę w tabeli.

a) Dla poniższego algorytmu dane stanowi skończony ciąg liczbowy zawierający co najmniej jedną liczbę:

1. $i := 0$
2. $wynik := 0$
3. dopóki nie przetworzono wszystkich liczb w ciągu wykonuj:
   i. $x :=$ kolejna liczba
   ii. $wynik := (i*wynik+x)/(i+1)$
   iii. $i := i+1$
4. wypisz wynik

Uwaga: „:=” oznacza instrukcję przypisania.

Wynikiem działania tego algorytmu jest

	P	F
suma podanych liczb.		
średnia arytmetyczna podanych liczb.		
średnia geometryczna podanych liczb.		

b) Poszukując numeru telefonu w książce telefonicznej wiele osób korzysta z następującego algorytmu: otwieramy książkę mniej więcej w połowie. Jeśli szukane nazwisko w kolejności alfabetycznej jest wcześniej niż nazwisko, na które trafiliśmy, otwieramy książkę w połowie, licząc od początku do miejsca, w którym się znajdujemy. W przeciwnym przypadku bierzemy pod uwagę drugą połowę książki. Postępujemy podobnie dla tej części książki, którą wybraliśmy, aż do momentu, kiedy jesteśmy blisko szukanego nazwiska. Wtedy wystarczy już przejrzeć kilka stron. Ten sposób postępowania jest zastosowaniem w praktyce strategii

	P	F
dziel i zwyciężaj.		
zachłannej.		
porządkowania ciągu elementów.		

c) Urządzenie, które pobiera dane cyfrowe z komputera i zamienia je na sygnały analogowe przesyłane w sieci telefonicznej to

	P	F
karta sieciowa.		
router.		
modem.		

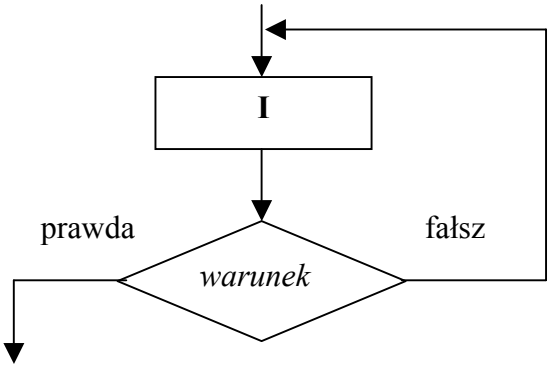
d) Zapis $1010_{(p)}$ oznacza, że 1010 jest zapisem pewnej liczby w systemie pozycyjnym o podstawie *p*. Zaznacz, która z poniższych równości jest prawdziwa:

	P	F
$1010_{(2)} = 10_{(10)}$		
$12_{(10)} = 1110_{(2)}$		
$67_{(10)} = 1000011_{(2)}$		

e) Kod ASCII znaku zero wynosi 48, a kodem małej litery „a” jest 97.

	P	F
Kodem znaku „3” jest liczba $00110100_{(2)}$.		
Kodem znaku „4” jest liczba $01100000_{(2)}$.		
Kodem małej litery „f” jest liczba $01100110_{(2)}$.		

f) Poniższy schemat blokowy opisuje instrukcję powtarzania, w której



	P	F
liczba powtórzeń instrukcji **I** nie zależy od warunku *warunek*.		
instrukcja **I** jest wykonywana co najmniej raz.		
jeśli *warunek* **nie jest** spełniony, to następuje zakończenie powtarzania.		

g) Do szyfrowania informacji służy

	P	F
algorytm RSA.		
algorytm Euklidesa.		
algorytm Hornera.		

h) Adresy IP składają się z czterech liczb z zakresu od 0 do 255, które zapisuje się oddzielone kropkami, np. 130.11.121.94. Pierwsza z liczb zapisana binarnie na ośmiu bitach pozwala określić, do jakiej klasy należy adres. Adresy klasy B mają na dwóch pierwszych bitach (licząc od lewej strony) wartości odpowiednio 1 i 0. Adresy klasy C mają na pierwszych trzech pozycjach wartości 1, 1 i 0.

	P	F
Adres 128.12.67.90 należy do klasy B.		
Adres 191.12.56.1 należy do klasy C.		
Adres 192.14.56.10 należy do klasy B.		

i) Skrótem nazwy protokołu sieciowego jest

	P	F
FTP.		
SSH.		
OSI.		

j) Plik graficzny zawiera obrazek o rozmiarach 1024 na 768 pikseli zapisany z użyciem 256 kolorów. Do zapisania tego pliku (bez użycia kompresji) potrzebne jest

	P	F
786432 bitów.		
786432 bajtów.		
786432 kilobajtów.		

k) Nazwą nośnika pamięci zewnętrznej jest

	P	F
płyta CD.		
pamięć flash.		
pamięć cache.		

l) Asymetryczne metody szyfrowania wymagają

	P	F
używania takich samych kluczy do szyfrowania i deszyfrowania wiadomości.		
używania różnych kluczy do szyfrowania i deszyfrowania wiadomości.		
ujawniania klucza służącego do szyfrowania.		

Sprawdzane umiejętności

W zadaniu były badane umiejętności z I obszaru standardów. Zdający:

- zna komputerową reprezentację znaków, liczb, tekstów i obrazów
- zna rolę, funkcję i zasady pracy sprzętu komputerowego oraz sieci komputerowej
- zna klasyczne algorytmy
- zna metody szyfrowania informacji

Rozwiązywalność zadania

61%, przy czym

a) 55%
b) 65%
c) 68%
d) 66%
e) 63%
f) 89%
g) 66%
h) 62%
i) 34%
j) 68%
k) 71%
l) 24%

Typowe poprawne odpowiedzi zdających

a) FPF
b) PFF
c) FFP
d) PFP
e) FFP
f) FPF
g) PFF
h) PFF
i) PPF
j) FPF
k) PPF
l) FPP

Najczęściej powtarzające się błędy
W punkcie b) zdający błędnie, jako prawidłową, podawali ostatnią odpowiedź: *porządkowania ciągu elementów*. Mieli również problemy z punktem g), w którym zaznaczali, że algorytm Hornera służy do szyfrowania informacji. W punkcie i) pomijali prawdziwą odpowiedź: *SSH*, co świadczy o nieznajomości protokołu zdalnej sesji SSH, umożliwiającego szyfrowanie połączenia między klientem a serwerem. W punkcie k) jako nośnika pamięci zewnętrznej nie uznawali pamięci flash, zaś w punkcie j) zaznaczali jako poprawną więcej niż jedną odpowiedź.

Komentarz
Zadanie okazało się umiarkowanie trudne dla zdających, łatwe były punkty f) i k), natomiast najtrudniejsze – punkty l) oraz i). Zadanie miało charakter testu wyboru sprawdzającego znajomość i rozumienie zagadnień z zakresu ogólnej wiedzy informatycznej. Wyniki uzyskane przez maturzystów były różnorodne, co wskazuje na duże zróżnicowanie poziomu ich wiedzy. Charakterystyczne, iż rezultaty uzyskane przez zdających wyraźnie były uzależnione od szkół, które kończyli. W niektórych szkołach zdający doskonale radzili sobie z tym zadaniem, w innych wyniki były znacząco niższe.

CZĘŚĆ II

Zadanie 4. Wybory (*20 pkt*)

W Infolandii przeprowadzono wybory parlamentarne. Do przydzielania mandatów zastosowano uproszczoną metodę d'Hondta opartą na obliczaniu współczynnika X:

$$X = \frac{v}{s+1},$$

gdzie:

v – to liczba głosów zdobytych przez dany komitet wyborczy w wyborach,

s – to liczba mandatów przydzielonych komitetowi do tej pory.

W każdym okręgu wyborczym mandaty przydziela się w następujący sposób: dopóki wszystkie mandaty nie zostaną przydzielone, dla każdego ugrupowania obliczany jest współczynnik X. W danym kroku algorytmu mandat otrzymuje ten komitet wyborczy, który ma największą wartość współczynnika X. W naszym zadaniu współczynniki X dla poszczególnych komitetów są różne w każdej fazie obliczeń.

Przykład

Załóżmy, że mamy zarejestrowane 3 komitety wyborcze: A, B i C, które otrzymały kolejno 950, 350 i 500 głosów w danym okręgu, a do obsadzenia jest 5 mandatów. W kolejnych krokach algorytmu mandaty przydzielamy na podstawie obliczonych współczynników dla poszczególnych komitetów wyborczych:

Krok algorytmu	Komitet A	Komitet B	Komitet C	Kto otrzymuje mandat?
1	$s=0$ $X=\frac{950}{1}=950$	$s=0$ $X=\frac{350}{1}=350$	$s=0$ $X=\frac{500}{1}=500$	A
2	$s=1$ $X=\frac{950}{2}=475$	$s=0$ $X=\frac{350}{1}=350$	$s=0$ $X=\frac{500}{1}=500$	C
3	$s=1$ $X=\frac{950}{2}=475$	$s=0$ $X=\frac{350}{1}=350$	$s=1$ $X=\frac{500}{2}=250$	A
4	$s=2$ $X=\frac{950}{3}=316{,}67$	$s=0$ $X=\frac{350}{1}=350$	$s=1$ $X=\frac{500}{2}=250$	B
5	$s=2$ $X=\frac{950}{3}=316{,}67$	$s=1$ $X=\frac{350}{2}=175$	$s=1$ $X=\frac{500}{2}=250$	A

Mandaty przypadają komitetom (kolejno) A, C, A, B, A. Zatem 3 mandaty zdobędzie komitet A, a po 1 mandacie komitety B i C.

a) Wybory odbyły się w 20 okręgach wyborczych. W parlamencie Infolandii ma zasiąść 350 posłów, z 6 różnych komitetów wyborczych o nazwach A, B, C, D, E, F.

Plik `dane.txt` zawiera dane dotyczące przeprowadzonych wyborów w podziale na okręgi. W każdym wierszu pliku znajduje się 7 liczb oddzielonych znakami odstępu: pierwsze sześć określają liczby oddanych ważnych głosów na kolejne komitety wyborcze (w kolejności A, B, C, D, E, F), a ostatnia oznacza **liczbę mandatów** do podziału w danym okręgu. Pierwszy wiersz zawiera dane dla okręgu wyborczego nr 1, drugi wiersz zawiera dane dla okręgu wyborczego nr 2, itd.

Przykład

```
325 155 200 248 311 69 15
478 198 321 487 54 14 18
```

Odpowiedzi do poniższych podpunktów umieść w pliku tekstowym `wybory.txt`. Odpowiedź do każdego podpunktu poprzedź cyfrą oznaczającą podpunkt.

1. Podaj, ile głosów łącznie otrzymał każdy z komitetów.
2. Podaj numery okręgów, w których łącznie na wszystkie komitety wyborcze oddano najwięcej i najmniej głosów.
3. Podaj liczby mandatów uzyskanych przez komitety A, B, C, D, E, F w okręgu 6.
4. Podaj, ile mandatów uzyskał każdy z komitetów A, B, C, D, E, F w całym parlamencie.

b) Dla danych z pliku `dane.txt` wykonaj wykres prezentujący procentowy rozkład liczby głosów oddanych w całej Infolandii na poszczególne komitety wyborcze. Pamiętaj o prawidłowym i czytelnym opisie wykresu.

Do oceny oddajesz plik(i) o nazwie(ach) .....,
tu wpisz nazwę(y) pliku(ów)
zawierający(e) komputerową(e) realizację(e) Twoich obliczeń, plik tekstowy `wybory.txt`, zawierający odpowiedzi do podpunktów zadania a) oraz plik o nazwie
....., zawierający wykres do zadania b).
tu wpisz nazwę pliku

Sprawdzane umiejętności
W zadaniu była badana umiejętność z II i III obszaru standardów: Zdający:
- dobiera właściwy program do rozwiązania zadania
- formułuje sytuację problemową i przystępuje do rozwiązania problemu w sposób planowy: wydziela podproblemy i wskazuje zależności między nimi, projektuje metody (algorytmy) rozwiązania zadania
- formułuje informatyczne rozwiązanie problemu przez odpowiedni dobór struktury danych oraz algorytmu i realizuje je w wybranym języku programowania (bądź w programie narzędziowym)

- stosuje narzędzia i techniki informatyczne do modelowania symulacji
- tworzy zestawienie wyników
- wykonuje analizę statystyczną różnych procesów z życia codziennego
- posługuje się programem użytkowym do graficznej prezentacji procentowego rozkładu sumy głosów oddanych we wszystkich okręgach

Rozwiązywalność zadania

38%, przy czym

a) 32%

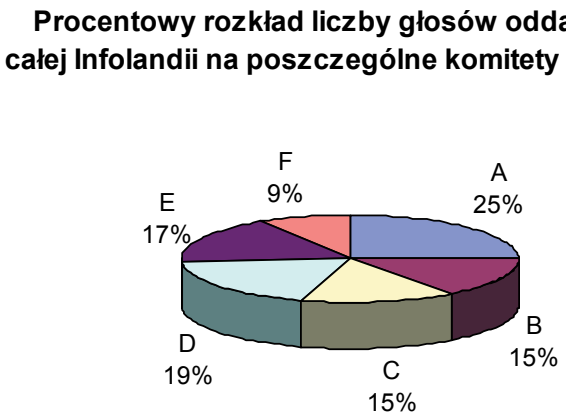
b) 60%

Typowe poprawne odpowiedzi zdających

a)

```
1
A-6331
B-3801
C-3866
D-4941
E-4351
F-2281
2
min: okręg nr 4
max: okręg nr 2
3
A-9
B-1
C-4
D-4
E-4
F-2
4
A - 93
B - 50
C - 50
D - 69
E - 62
F - 26
```

b)



Większość zdających do rozwiązania zadania w punkcie a) wybrała napisanie programu w wybranym przez siebie języku programowania, np. takie jak na następnych stronach – ale były też rozwiązania pracochłonne w arkuszu kalkulacyjnym (z wykorzystaniem odpowiednich formuł i ich powielaniem). Część zdających podała prawidłowe odpowiedzi tylko w pliku tekstowym, ale nie załączyła komputerowej realizacji rozwiązania i nie otrzymała punktów.

```
program Wybory;

const
 l_okr = 20;  {Liczba okręgów}
 l_kom = 6;   {Liczba komitetów}

 var
  wyn        : text;                                    {Plik z wynikami}
  glosy      : array[1..l_okr,1..l_kom] of integer;     {Liczby głosów
                                                         oddanych w
                                                         poszczególnych
                                                         okręgach}
  l_mandatow : array[1..l_okr] of integer;              {Liczby mandatów do
                                                         rozdzielenia w
                                                         poszczególnych
                                                         okręgach}
  mandat     : array[1..l_okr,1..l_kom] of integer;     {Liczby mandatów
                                                         przydzielonych w
                                                         poszczególnych
                                                         okręgach}
  komitety   : array[1..l_kom] of char;                 {Nazwy poszczególnych
                                                         komitetów}

procedure wczytaj_dane; {Wczytanie danych do tablicy}
 var
  nazwa_komitetu : char;
  dane           : text;      {Plik z danymi}
  i              : 1..l_okr;  {Indeks}
  j              : 1..l_kom;  {Indeks}
  k              : 0..l_kom;  {Indeks}
begin
  assign(dane, 'dane.txt');
  reset(dane);

  k := 0;
  for nazwa_komitetu:='A' to 'F' do
  begin
    k := k + 1;
    komitety[k] := nazwa_komitetu;
  end;

  for i := 1 to l_okr do
   begin
    for j := 1 to l_kom do
      read(dane, glosy[i,j]);
    read(dane, l_mandatow[i]);
    readln(dane);
   end;

  close(dane);
end;

procedure mandaty;
 var
  ktory   : integer;   {Który komitet ma dotąd najwyższy współczynnik v}
  max     : real;      {Ile wynosi dotąd najwyższy współczynnik v}
  v       : real;      {Wartość współczynnika v}
  i       : 1..l_okr;  {Indeks}
  j       : 1..l_kom;  {Indeks}
  k       : integer;   {Indeks}

begin
  for i := 1 to l_okr do
    for j := 1 to l_kom do
      mandat[i,j] := 0;

  for i := 1 to l_okr do
   begin
    for k := 1 to l_mandatow[i] do {przydzielenie kolejnego mandatu}
```

```
     begin
       max   := 0;
       ktory := 1;
       for j := 1 to l_kom do
        begin
         v := glosy[i,j] / (mandat[i,j]+1);
         if max < v then
          begin
           max   := v;
           ktory := j;
          end;
        end; {for j}
      mandat[i,ktory] := mandat[i,ktory] + 1;
     end; {for k}
   end; {for i}
end; {mandaty}

procedure pkt1;
 var
  suma : integer;   {Liczba głosów bieżącego komitetu}
  i    : 1..l_okr;  {Indeks}
  j    : 1..l_kom;  {Indeks}

begin
  writeln(wyn,'1');
  for j := 1 to l_kom do
   begin
    suma := 0;
    for i := 1 to l_okr do
     suma := suma + glosy[i,j];
    writeln(wyn, komitety[j], '-', suma);
   end;
end;

procedure pkt2;
 var
  suma      : integer;   {Liczba głosów w bieżącym  okręgu}
  max, min  : integer;   {Największa i najmniejsza dotąd liczba głosów w
okręgu}
  ktory_min,
  ktory_max : 0..l_okr;  {Numery okręgów o najmniejszej i największej
liczbie głosów}
  i         : 1..l_okr;  {Indeks}
  j         : 1..l_kom;  {Indeks}
begin
  writeln(wyn, '2');
  max := 0;
  min := maxint;
  ktory_min := 0;
  ktory_max := 0;
  for i := 1 to l_okr do
  begin
    suma := 0;
    for j := 1 to l_kom do
      suma := suma + glosy[i,j];
    if suma > max then
       begin
        max := suma;
        ktory_max := i;
       end;
    if suma < min then
       begin
        min := suma;
        ktory_min := i;
       end;
  end;
  writeln(wyn, 'min: okreg nr ', ktory_min);
  writeln(wyn, 'max: okreg nr ', ktory_max);
end;
```

```
procedure pkt3;
 const
  nr_okr = 6; {Dla tego komitetu liczymy liczbę mandatów}
 var
  j : 1..l_kom;  {Indeks}
begin
  writeln(wyn, '3');
  for j := 1 to l_kom do
    writeln(wyn, komitety[j], '-', mandat[nr_okr,j]);
end;

procedure pkt4;
 var
  suma : integer;   {Liczba mandatów bieżącego komitetu}
  i    : 1..l_okr;  {Indeks}
  j    : 1..l_kom;  {Indeks}
begin
  writeln(wyn, '4');
  for j := 1 to l_kom do
   begin
    suma := 0;
    for i := 1 to l_okr do
      suma := suma + mandat[i,j];
    writeln(wyn, komitety[j], ' - ', suma);
   end;
end;

begin {Program}
  wczytaj_dane;              {Wczytanie danych }

  mandaty;                   {Obliczenie liczby mandatów}

  assign(wyn, 'wybory.txt'); {Utworzenie}
  rewrite(wyn);              {pliku wynikowego}

  {Wyznaczanie i zapisywanie odpowiedzi do poszczególnych podpunktów}
  pkt1;
  pkt2;
  pkt3;
  pkt4;

  close(wyn);
end.
```

Najczęściej powtarzające się błędy

Zdający mieli największe trudności z podpunktami a3 i a4, w których należało zastosować podany algorytm do przydzielania mandatów; niektórzy podawali błędne odpowiedzi do tych podpunktów, które wynikały z zupełnego niezrozumienia opisanej metody.

W podpunkcie a2, niektórzy zdający podawali nazwy komitetów zamiast numerów okręgów lub numery wierszy, jeśli zadanie rozwiązywali w arkuszu. Zdarzało się, że kolumna z liczbą mandatów do podziału na dany okręg była traktowana jako liczba głosów oddanych na siódmy komitet.

W podpunkcie a4 niektórzy zdający, zamiast liczyć mandaty w każdym okręgu i później je zsumować, najpierw sumowali liczbę głosów (we wszystkich okręgach) i na tej podstawie wyznaczali liczbę mandatów stosując metodę d'Hondta. Takie rozwiązanie było niezgodne z treścią zadania, gdzie było napisane, że mandaty przydziela się osobno w każdym okręgu.

W punkcie b) częstym błędem było tworzenie wykresu na podstawie danych z całej tabeli z danymi szczegółowymi z rozbiciem na okręgi (zamiast sum obliczonych w podpunkcie a1) i stosowanie do nich bezpośrednio wykresu kolumnowego. Zdarzały się również wykresy z błędem polegającym na tym, że wykresy były utworzone dla rozkładu głosów oddanych w poszczególnych okręgach wyborczych zamiast dla rozkładu głosów oddanych na poszczególne komitety wyborcze.

Komentarz
Zadanie okazało się trudne, przy czym punkt b) – umiarkowanie trudny. Zdający najczęściej wykorzystywali arkusz kalkulacyjny do wygenerowania odpowiedzi do podpunktów a1, a2 oraz punktu b), prawidłowo sumując głosy oddane na poszczególne komitety, wyszukując maksimum i minimum wśród sum głosów oddanych w poszczególnych okręgach wyborczych oraz tworząc wykres kołowy lub kolumnowy, pozwalający odczytać procentowy rozkład liczby głosów na poszczególne komitety wyborcze w całej Infolandii. Prawidłowe odpowiedzi do podpunktów a3 i a4 można uzyskać pisząc prosty program, obliczający liczbę mandatów w poszczególnych okręgach. Zdający wyraźnie nie umieli zastosować podanego algorytmu do przydzielania mandatów, niektórzy maturzyści udzielali odpowiedzi do punktów 3 i 4 wykazując się kompletnym niezrozumieniem całej opisanej procedury.

Zadanie 5. Hasła jednorazowe (*22 pkt*)

Infobank oferuje swoim klientom internetowe konta osobiste. Do korzystania z tych kont potrzebna jest lista haseł jednorazowych. Jesteś pracownikiem Wydziału Bezpieczeństwa. Wydział ten zajmuje się tworzeniem dla klientów list haseł jednorazowych na podstawie słów wcześniej wygenerowanych przez Wydział Statystyk, według ustalonych przez bank algorytmów.

Plik `slowa.txt`, zawiera 1000 słów o długościach nie większych niż 30 znaków. Każde słowo jest zapisane w osobnym wierszu i złożone jest z wielkich liter alfabetu angielskiego.

a) Na początku swojej działalności bank stosował następującą metodę generowania haseł. Hasłami jednorazowymi są zapisane od końca słowa wygenerowane przez Wydział Statystyk.

Przykład

Słowo	Utworzone hasło
KAJAK	KAJAK
EGZAMIN	NIMAZGE
MATURA	ARUTAM
KOMINIARZ	ZRAINIMOK

Dla danych z pliku `slowa.txt` utwórz listę haseł jednorazowych i umieść ją w pliku `hasla_a.txt` (każde hasło w osobnym wierszu). Dodatkowo w pliku `slowa_a.txt` podaj najdłuższe i najkrótsze hasła oraz ich długości.

Uwaga: Dla danych z pliku `slowa.txt` da się utworzyć tylko po jednym haśle o największej i najmniejszej długości. Ta uwaga ma też zastosowanie w podpunkcie b).

b) Metoda tworzenia haseł opisana w punkcie a) okazała się zawodna, dlatego Wydział Bezpieczeństwa postanowił zmienić algorytm.

Palindrom to takie słowo, które czytane od lewej do prawej i od prawej do lewej jest takie same.

Algorytm tworzenia hasła ze słowa `w`:

- wyznacz najdłuższe słowo w_1 takie, że w_1 jest początkiem słowa w oraz w_1 jest palindromem
- oznaczmy $w=w_1w_2$
- hasło powstaje przez złączenie dwóch słów: w_2 zapisanego od prawej do lewej i w.

Uwaga: Jeśli w jest palindromem, to $w=w_1$, a słowo w_2 jest puste (nie zawiera żadnych znaków).

Przykład

Słowo	Najdłuższy palindrom rozpoczynający słowo	Utworzone hasło
KAJAK	KAJAK	KAJAK
KAJAKARSTWO	KAJAK	OWTSRAKAJAKARSTWO
MAMA	MAM	AMAMA
KAKTUS	KAK	SUTKAKTUS
WANNA	W	ANNAWANNA
EGZAMIN	E	NIMAZGEGZAMIN

Dla danych znajdujących się w pliku slowa.txt, utwórz listę haseł według nowego algorytmu. Utworzone hasła jednorazowe umieść w pliku hasla_b.txt (każde hasło w osobnym wierszu).

Dodatkowo odpowiedzi do poniższych podpunktów umieść w pliku tekstowym slowa_b.txt. Odpowiedź do każdego podpunktu poprzedź cyfrą oznaczającą podpunkt.

1. Podaj wszystkie hasła o długości 12.
2. Podaj najdłuższe i najkrótsze hasło.
3. Podaj sumę długości wszystkich haseł.

Do oceny oddajesz plik(i) o nazwie(ach) ....., zawierający(e) komputerową(e) realizację(e) Twojego rozwiązania, pliki tekstowe hasla_a.txt, slowa_a.txt zawierające odpowiedzi do zadania a) oraz pliki tekstowe hasla_b.txt, slowa_b.txt zawierające odpowiedzi do zadania b).

tu wpisz nazwę(y) pliku(ów)

Sprawdzane umiejętności

W zadaniu były badane umiejętności z III obszaru standardów. Zdający:

- dobiera struktury danych do przetwarzanych informacji korzystając przy tym z podstawowych struktur danych (znaki, ciągi znaków, pliki) oraz układa algorytmy dla zadanych problemów i implementuje je w wybranym języku programowania (bądź innym narzędziu),
- tworzy zestawienie wyników
- wykorzystuje metody informatyki (konstrukcje algorytmiczne, klasyczne algorytmy) do rozwiązania problemów

Rozwiązywalność zadania

16%, przy czym

a) 28%

b) 9%

Typowe poprawne odpowiedzi zdających

Zadanie programistyczne, sprawdzające umiejętność zapisu i zastosowania klasycznych algorytmów nauczanych w szkołach z wykorzystaniem podstawowych struktur danych. Jednak w punkcie a) można było spotkać rozwiązania zrealizowane za pomocą arkusza.

a)
Ze względu na obszerność nie podajemy zawartości pliku `hasla_a.txt`, można ją znaleźć na stronie www.cke.edu.pl. Poprawna zawartość pliku `slowa_a.txt`:

```
EHJA 4
DCBAJAHHJIGABCDCCDCBAGIJHHAJ 28
```

Poprawny przykładowy program:

```
program Zadanie_5a;

var
 dane      : text;    {Plik z danymi}
 wynik1    : text;    {Plik z zaszyfrowanymi słowami}
 wynik2    : text;    {Plik z policzonymi odpowiedziami}
 slowo     : string;  {Kolejne słowo}
 slowo_min : string;  {Dotąd najkrótsze słowo}
 slowo_max : string;  {Dotąd najdłuższe słowo}
 dl        : integer; {Długosc slowo}
 min       : integer; {Długość slowo_min}
 max       : integer; {Długość slowo_max}


procedure odwroc(var slowo: string);
 {Odwaca kolejność znaków w slowo}
 var
  l, p   : integer; {Indeksy}
  pom    : char;    {Pomocnicza}
begin
  l     := 1;
  p     := length(slowo);
  while l < p do
   begin
    pom      := slowo[l];
    slowo[l] := slowo[p];
    slowo[p] := pom;
    l := l + 1;
    p := p - 1;
   end; {while}
end; {Odwróć}

begin
  assign(dane, 'slowa.txt');
  reset(dane);
  assign(wynik1, 'hasla_a.txt');
  rewrite(wynik1);
  assign(wynik2, 'slowa_a.txt');
  rewrite(wynik2);

  min := MaxInt;
  max := 0;
  while not eof(dane) do
   begin
    readln(dane, slowo);
    odwroc(slowo);
    dl := length(slowo);
    if dl < min then
     begin
      slowo_min := slowo;
      min       := dl;
     end;
    if dl > max then
     begin
      slowo_max := slowo;
      max       := dl;
     end;
    writeln(wynik1, slowo);
   end; {while}
```

```
  close(dane);
  writeln(wynik2, slowo_min, ' ',min);
  writeln(wynik2, slowo_max, ' ',max);
  close(wynik1);
  close(wynik2)
end.
```

b)
Ze względu na obszerność nie podajemy zawartości pliku `hasla_b.txt`, można ją znaleźć na stronie www.cke.edu.pl. Poprawna zawartość pliku `slowa_b.txt`:

```
1
BDJBFBBFBJDB
DFDAHEEHADFD
IHBIBCCBIBHI
FABBEAAEBBAF
2
OKOOKO
AJBEIKKFIIGIIJBAHADIBCIDHDICBIDAHABJIIGIIFKKIEBJA
3
27731
```

Poprawny przykładowy program:

```
program Zadanie_5b;

const
 szuk_dl = 12;  {Szukamy słów tej długości}

var
 dane      : text;    {Plik z danymi}
 wynik1    : text;    {Plik z zaszyfrowanymi słowami}
 wynik2    : text;    {Plik z policzonymi odpowiedziami}
 slowo     : string;  {Kolejne słowo}
 slowo_min : string;  {Dotąd najkrótsze słowo}
 slowo_max : string;  {Dotąd najdłuższe słowo}
 dl        : integer; {Długość slowo}
 min       : integer; {Długość slowo_min}
 max       : integer; {Długość slowo_max}
 suma_dl   : integer; {Suma długości wszystkich zaszyfrowanych słów}

function odwroc(slowo: string): string;
 {Odwraca kolejność znaków w slowo}
 var
  l, p   : integer; {Indeksy}
  pom    : char;    {Pomocnicza}
begin
  l     := 1;
  p     := length(slowo);
  while l < p do
   begin
    pom      := slowo[l];
    slowo[l] := slowo[p];
    slowo[p] := pom;
    l := l + 1;
    p := p - 1;
   end; {while}
  odwroc := slowo;
end; {Odwróć}

function czy_palindrom(poczatek, koniec: integer; slowo: string): boolean;
 {Sprawdza czy slowo[poczatek..koniec] jest palindromem}
 var
  odp : boolean; {Wynik funkcji}
begin
  odp := true;
  while (poczatek < koniec) and odp do
```

```
   begin
    if slowo[poczatek] <> slowo[koniec] then
      odp := false;
    poczatek := poczatek + 1;
    koniec   := koniec - 1;
   end; {while}
  czy_palindrom := odp;
end; {Czy_palindrom}

function szyfruj(slowo: string): string;
 {Szyfruje zadane słowo}
 var
  j : integer;
begin
  j := length(slowo);
  while not czy_palindrom(1, j, slowo) do
    j := j - 1;
  szyfruj := odwroc(slowo) + copy(slowo, j+1, length(slowo)-j);
end; {Szyfruj}

begin {Program}
  assign(dane, 'slowa.txt');
  reset(dane);
  assign(wynik1, 'hasla_b.txt');
  rewrite(wynik1);
  assign(wynik2, 'slowa_b.txt');
  rewrite(wynik2);

  writeln(wynik2, '1');

  min       := MaxInt;
  max       := 0;
  slowo_min := '';
  slowo_max := '';
  suma_dl   := 0;
  while not eof(dane) do
   begin
    readln(dane, slowo);
    slowo := szyfruj(slowo);
    dl := length(slowo);
    if dl = szuk_dl then
        writeln(wynik2, slowo);
    if dl < min then
       begin
        slowo_min := slowo;
        min       := dl
       end;
    if dl > max then
     begin
        slowo_max := slowo;
        max       := dl
       end;
    suma_dl := suma_dl + dl;
    writeln(wynik1, slowo);
   end; {while}

  close(dane);
  close(wynik1);

  writeln(wynik2, '2');
  writeln(wynik2, slowo_min);
  writeln(wynik2, slowo_max);
  writeln(wynik2, '3');
  writeln(wynik2, suma_dl);

  close(wynik2);
end.
```

Najczęściej powtarzające się błędy
Częstym błędem w punkcie a) było wyszukiwanie najdłuższego i najkrótszego słowa w pliku źródłowym `slowa.txt`, czyli przed przekształceniem słów do postaci haseł.
W punkcie b), którego rozwiązania podjęło się zdecydowanie mniej zdających, spotkać można było błędy w niektórych wygenerowanych hasłach, wynikające z niepoprawnych warunków brzegowych w programie przetwarzającym napisy oryginalne do postaci haseł wynikowych.

Komentarz
Zadanie okazało się bardzo trudne dla zdających. Jest to typowe zadanie programistyczne – wymagało *sprawności* w programowaniu i dlatego bardzo mało zdających podejmowało się jego rozwiązania, szczególnie punktu b). Podobnie jak w pierwszym arkuszu, wynik tego zadania potwierdza, że maturzyści mają problemy z algorytmiką i zapisywaniem algorytmów w postaci poprawnych programów komputerowych. Należy tutaj zwrócić uwagę, że są to kluczowe umiejętności informatyczne.

Zadanie 6. Wypadki (*18 pkt*)

Towarzystwo ubezpieczeniowe posiada w swoim rejestrze pojazdów następujące dane o samochodach osobowych: **numer rejestracyjny**, **marka**, **rok produkcji, numer PESEL właściciela** oraz dane o właścicielach pojazdów: **imię**, **nazwisko**, **numer PESEL**, **typ miejscowości**. Ponadto gromadzi informacje o wypadkach spowodowanych przez ubezpieczonych właścicieli samochodów, aby na tej podstawie ustalać składki ubezpieczenia.

W kolejnych 700 wierszach pliku `auta.txt` znajdują się następujące dane dotyczące samochodów: numer rejestracyjny, marka, rok produkcji, numer PESEL właściciela pojazdu.

Przykład

```
BAU1876 skoda 1998 59042500616
BAU3353 renault 1999 54010520609
```

W kolejnych 689 wierszach pliku `osoby.txt` znajdują się następujące dane: numer PESEL, imię, nazwisko, typ miejscowości. Przyjęto następujące oznaczenia typów miejscowości: A – duże miasto, B – średnie miasto, C – małe miasto i D – wieś.

Przykład

```
46073182890 Kornel Henrykowski A
46080423256 Jan Bugajski B
```

W kolejnych 500 wierszach pliku `wypadki.txt` znajdują się następujące dane: numer identyfikacyjny wypadku, data wypadku, numer rejestracyjny samochodu, wysokość straty, którą pokryło towarzystwo ubezpieczeniowe.

Przykład

```
1 1996-01-03 BL24933 10453,00
2 1997-10-14 GCH9779 673,00
3 2002-03-24 NWE4941 8276,00
```

Separatorem oddzielającym sąsiednie elementy w powyższych plikach jest znak odstępu.
Odpowiedzi do poniższych podpunktów umieść w pliku tekstowym `odp.txt`. Odpowiedź do każdego podpunktu poprzedź literą oznaczającą podpunkt.

a) Podaj, ilu właścicieli samochodów miało co najmniej jeden wypadek. Uwaga: Właściciela odnotowanego w kilku wypadkach liczymy jeden raz.

b) Podaj numer rejestracyjny samochodu oraz imię i nazwisko właściciela, któremu wypłacono największą kwotę odszkodowania oraz jej wysokość.

c) Podaj sumy odszkodowań, jakie wypłaciło towarzystwo ubezpieczeniowe w roku 2006 oraz w roku 2007.

d) Podaj markę samochodu, która została odnotowana w największej liczbie wypadków oraz liczbę wypadków, w których samochody tej marki były odnotowane. Jeśli pewien samochód był odnotowany w kilku wypadkach, to liczymy go tyle razy, w ilu wypadkach brał udział.

e) Podaj liczby wypadków z udziałem właścicieli z małego, średniego i dużego miasta oraz ze wsi (oddzielnie dla każdego typu miejscowości).

Do oceny oddajesz plik(i) o nazwie(ach) ....., zawierający(e) komputerową(e) realizację(e) Twoich rozwiązań oraz plik `odp.txt`, zawierający odpowiedzi na pytania z podpunktów a) – e). Każda odpowiedź powinna być poprzedzona odpowiednią literą oznaczającą podpunkt.

tu wpisz nazwę(y) pliku(ów)

Sprawdzane umiejętności
W zadaniu były badane umiejętności z II i III obszaru standardów. Zdający:

- projektuje i tworzy strukturę bazy danych będącą reprezentacją zbioru informacji i relacji między nimi
- stosuje metody wyszukiwania i przetwarzania i informacji w relacyjnych bazach danych.

Rozwiązywalność zadania
38%, przy czym

a) 29%
b) 51%
c) 39%
d) 37%
e) 34%

Typowe poprawne odpowiedzi zdających
Dominowały rozwiązania w programie bazodanowym, ale spotkać można było także prace, gdzie użyto arkusza kalkulacyjnego (z użyciem filtrowania lub formuł). Zdarzały się prace, gdzie odpowiedzi na niektóre pytania były udzielane na podstawie arkusza, a na inne w bazie danych.

Poprawna zawartość pliku `odp.txt`:

```
a) 326

b) PKR9139  Tomasz      Misiak      10486

c)

2006 194741
2007 227446

d)  fiat 145

e)

A     138
B     115
C     133
D     114
```

Poprawne kwerendy do poszczególnych punktów zadania:

a)
```
SELECT Count(*) AS Liczba
FROM [SELECT DISTINCT pesel
FROM Wypadki INNER JOIN Auta ON Wypadki.[numer rejestracyjny] =Auta.[Numer
rejestracyjny]] AS T;
```

b)
```
SELECT TOP 1 Wypadki.[numer rejestracyjny], imię, nazwisko, [wysokość
straty]
FROM Osoby INNER JOIN (Wypadki INNER JOIN Auta ON Wypadki.[numer
rejestracyjny]=Auta.[Numer rejestracyjny]) ON Osoby.Pesel=Auta.Pesel
ORDER BY [wysokość straty] DESC;
```

c)
```
SELECT Sum(Wypadki.[wysokość straty]) AS [SumaOfwysokość straty],
year(Wypadki.[data wypadku]) AS rok
FROM Wypadki
WHERE (((Wypadki.[data wypadku])>=#1/1/2006# And (Wypadki.[data
wypadku])<=#12/31/2007#))
GROUP BY year(Wypadki.[data wypadku]);
```

d)
```
SELECT TOP 1 Auta.marka, Count(Wypadki.nr) AS PoliczOfnr
FROM Auta INNER JOIN Wypadki ON Auta.[Numer rejestracyjny]=Wypadki.[numer
rejestracyjny]
GROUP BY Auta.marka
ORDER BY Count(Wypadki.nr) DESC;
```

e)
```
SELECT Osoby.[Typ miejscowości], Count(Wypadki.[numer rejestracyjny]) AS
[PoliczOfnumer rejestracyjny]
FROM (Osoby INNER JOIN Auta ON Osoby.Pesel=Auta.Pesel) INNER JOIN Wypadki
ON Auta.[Numer rejestracyjny]=Wypadki.[numer rejestracyjny]
GROUP BY Osoby.[Typ miejscowości];
```

Najczęściej powtarzające się błędy

W wielu pracach źle były zbudowane relacje między tabelami, zaś w zapytaniach niepoprawnie formułowano warunki, bądź pomijano kryteria.

Największy problem sprawiał punkt a), zdający podawali liczbę wszystkich wypadków (500), zamiast liczby właścicieli, którzy mieli co najmniej jeden wypadek (brak grupowania rekordów).

W punkcie b) zdarzało się, że zdający znalazł poprawną największą kwotę, ale nie umiał jej

powiązać z właścicielem. W innych rozwiązaniach było na odwrót – brakowało kwoty, a podawano poprawnie numer rejestracyjny samochodu oraz imię i nazwisko właściciela. Błędy tego typu występowały przede wszystkim w rozwiązaniach tworzonych z użyciem arkusza kalkulacyjnego i wynikały zapewne z nieumiejętności powiązania ze sobą różnych tabel. W tym punkcie napotkano też odpowiedzi, dotyczące nie najwyższej kwoty odszkodowania, ale sumy wszystkich odszkodowań wypłaconych jednej osobie.
W punkcie c) zdarzało się, że podawana była łączna kwota odszkodowań za rok 2006 i 2007 zamiast dla każdego roku osobno.
W ostatnim punkcie błędne odpowiedzi wynikały z tego, że zdający mieli problemy z powiązaniem danych z plików `osoby.txt` i `wypadki.txt`.

Komentarz

Zadanie okazało się dość trudne dla zdających, pomimo, że jest klasycznym przykładem zadania sprawdzającego umiejętność przetwarzania danych w postaci prostej relacyjnej bazy danych (podobne zadania występowały w arkuszach egzaminacyjnych w poprzednich latach). Rozwiązanie wymagało łączenia tabel, sortowania, filtrowania i grupowania danych. Niewielki rozmiar analizowanych danych, prosty schemat tabel i tekstowa postać danych wejściowych dawały dużą swobodę wyboru narzędzi do rozwiązania tego zadania. Osoby rozwiązujące to zadanie realizowały je zarówno w programach narzędziowych do obsługi relacyjnych baz danych, w arkuszu kalkulacyjnym, jak również pisały programy w języku programowania. Największe kłopoty sprawiały zdającym następujące elementy:

- łączenie tabel
- formułowanie warunków i kryteriów w zapytaniach (problemy z interpretacją spójników).

Powyższe problemy były najczęstszymi powodami błędnych odpowiedzi. Na koniec trzeba zaznaczyć, że niewielka część zdających nie podjęła próby rozwiązania tego zadania. Czasami problemem dla sprawdzających prace egzaminatorów był brak pliku tekstowego z odpowiedziami lub brak odpowiedzi w pliku tekstowym do niektórych punktów – egzaminatorzy znajdowali prawidłowe wyniki przeglądając utworzone kwerendy lub pliki arkusza kalkulacyjnego.

PODSUMOWANIE

Podczas egzaminu maturalnego z informatyki sprawdzano umiejętności absolwentów w trzech obszarach standardów wymagań:

I. znajomości i rozumienia podstawowych pojęć, metod, narzędzi i procesów związanych z informatyką;
II. stosowania posiadanej wiedzy do rozwiązywania zadań teoretycznych i praktycznych;
III. stosowania metod informatycznych do rozwiązywania problemów.

Arkusze egzaminacyjne z informatyki składały się z różnych rodzajów zadań. Były zadania otwarte oraz zamknięte, zadania wymagające umiejętności programistycznych, konstruowania i analizowania algorytmów, posługiwania się programami narzędziowymi. Można je było rozwiązać różnymi metodami, przy pomocy różnych narzędzi. Decyzja wyboru metody i programu należała do zdającego, nie została narzucona „z góry".

Zadania były sformułowane precyzyjnie i krótko.

Analizując tegoroczne wyniki egzaminu maturalnego z informatyki warto zauważyć, że średni wynik (36%) jest prawie identyczny, jak wynik uzyskany przez zdających w ubiegłym roku i o kilkanaście procent wyższy niż w latach 2005-2006. Świadczy to o coraz lepszym przygotowaniu absolwentów do egzaminu z tego przedmiotu. Niemniej jednak, podobnie jak w poprzednich sesjach egzaminu, zdający mają najwięcej problemów z zadaniami dotyczącymi algorytmiki i programowania.