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

ZADANIE 1 – TO CHYBA BĘDZIE JEZIORO ŁABĘDZIE

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

Dana jest dwuwymiarowa tablica o stałym wymiarze 50×100 symulująca prostokątne jezioro. Na jeziorze tym znajduje się mama łabędź (oznaczona symbolem X), tata łabędź (oznaczony symbolem Y) i dzieci łabędzie (oznaczone liczbami naturalnymi od 1 do n).

Początkowe położenie łabędzi jest takie, że Y znajduje się w lewym-górnym rogu jeziora, pod nim znajduje się łabędziątko o numerze n, pod nim łabędziątko o numerze $n-1$, i tak dalej, aż do X.

Na przeciwległym brzegu jeziora, po jego środku, znajduje się gniazdo łabędzi (oznaczone symbolami g) o wymiarze 5×2. W pozostałych wolnych, ale jednocześnie nie sąsiadujących z brzegami, położeniem łabędzi, gniazdem i między sobą polach znajduje się n okruchów chleba (oznaczonych symbolem o).

Zadaniem rodziców jest doprowadzenie rodziny do gniazda po uprzednim jej nakarmieniu.

Rodzina porusza się zgodnie z wyborem jednej ze strzałek na klawiaturze (wciskać można dowolną strzałkę, ale działanie podjęte będzie tylko przy możliwym ruchu – np. pierwszy ruch to strzałka w dół lub strzałka w prawo, ogólnie możliwym ruchem jest przesunięcie się na wolne pole lub na pole, w którym położony jest okruch chleba). Pierwsza porusza się mama do miejsca, które wskazuje strzałka, za nią płyną koleje dzieci i tata, zajmując poprzednio zajęte miejsca swoich poprzedników.

Po natrafieniu na pole z chlebem, okruch taki zjadany jest przez pierwszego z kolei łabędzia, który nie jadł jeszcze chleba (w drugiej „rundzie” – nie jadł jeszcze drugi raz chleba, w trzeciej – nie jadł trzeci raz, itd.). Po zjedzeniu kawałka chleba w losowym wolnym i dozwolonym polu pojawia się kolejny okruch. Zjedzenie chleba przez łabędziątko sygnalizowane jest przez zwiększenie jego symbolu o 1. Rodzice doprowadzić mogą rodzinę do gniazda tylko wówczas, gdy każde z łabędziątek zje co najmniej po dwa okruchy chleba.

Podczas podróży może się tak zdarzyć, że mama łabędź nie będzie miała gdzie popłynąć. Może one wówczas zanurkować – sygnalizujemy to wciskając przycisk n oraz odpowiednią strzałkę, a następnie liczbę (np. $n, \rightarrow, 3$). Taka kombinacja klawiszy oznaczała będzie zanurkowanie i przepłyniecie pod wodą (wówczas symbole łabędzi znajdujące się pod wodą są niewidoczne) w określonym kierunku określoną liczbę pól. Oczywiście ruch taki musi być możliwy do wykonania. Manewr taki może być jednak wykonany jedynie raz na każde 5 posiadanych łabędziątek (dokładniej, dla $1 \leq n \leq 5$ – jeden możliwy manewr, dla $6 \leq n \leq 10$ – dwa możliwe manewry, itd.).

Po wykarmieniu wszystkich łabędziątek i dotarciu mamy do gniazda, reszta rodziny, bez udziału użytkownika przechodzi za mamą do gniazda.

Jeżeli dojdzie do sytuacji, w której nie będzie dozwolonych ruchów, podróż zakończy się niepowodzeniem.

Napisz program, który dla zadanego przez użytkownika $1 \leq n \leq 20$ (liczba łabędziątek) przeprowadzał będzie wizualizację wyżej opisanej podróży (formę wizualizacji pozostawiamy w gestii rozwiązującego to zadanie).

ZADANIE 2 – GRA W WOJNĘ

Zadanie zaproponował: dr inż. Mariusz Pleszczyński, Wydział Matematyki Stosowanej, Politechnika Śląska (zadanie finałowe 2023)

Jedną z pierwszych gier karcianych, jaką poznaje się w początkowym zetknięciu z kartami, jest gra w wojnę. W grę tę można grać w więcej niż w dwie osoby, my jednak ograniczymy się do dwóch graczy.

W grze (w naszym przypadku) tej używa się standardowej talii złożonej z 24 kart. Każdemu z graczy rozdaje się (w naszym przypadku – losuje się) 12 kart. Karty te ułożone są koszulkami do góry (obrazki są niewidoczne).

Każdy z graczy odwraca pierwszą z góry swojego stosu kartę. Ten z graczy, którego karta jest wyższa (hierarchia kart od najwyższej: as, król, dama, walet, dziesiątka, dziewiątka, przy czym ich kolor nie ma znaczenia), zabiera pod spód swojego stosu (koszulkami do góry) najpierw kartę przeciwnika, a potem swoją.

W przypadku, gdy wyłożone karty są tej samej ważności, mamy do czynienia z wojną: każdy z graczy wykłada z góry swojego stosu jedną kartę i dokłada jeszcze jedną. Ta druga karta decyduje o tym, kto wygrywa wojnę i zabiera karty przeciwnika.

Jeśli wojna nadal jest nierozstrzygnięta, to wojna trwa dalej (do rozstrzygnięcia) i gracze ponownie odkrywają kolejno dwie kraty, z których druga decyduje o zwycięstwie. Zwycięzca wojny zabiera karty przeciwnika wkładając je kolejno koszulkami do góry – najpierw kolejne karty z odkrytego stosu przeciwnika, a potem ze swojego.

Gra kończy się wówczas, gdy jeden z graczy straci wszystkie swoje karty.

Podczas rozgrywki może się tak zdarzyć, że po rozpoczęciu wojny jednemu z graczy może zabraknąć kart do jej zakończenia (wojna rozpoczęła się po wyłożeniu ostatniej lub przedostatniej karty). W takim przypadku gracz, który ma więcej kart wykłada w odpowiednim momencie ze swojego stosu kartę na stos przeciwnika (zakładamy, że przeciwnik dostaje potrzebną mu kartę jako pierwszy).

Napisz program, który wizualizował będzie taką rozgrywkę (formę wizualizacji pozostawiamy w gestii rozwiązującego to zadanie).

Czy istnieje rozdanie, w którym gra jest nierozstrzygalna? Jeśli tak, podaj przykład takiego rozdania. Jeśli nie – uzasadnij dlaczego.

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

ZADANIE 3 – SKŁADANIE PERMUTACJI

Zadanie zaproponował: dr inż. Mariusz Pleszczyński, Wydział Matematyki Stosowanej, Politechnika Śląska (rozszerzenie zadania 4 z finału poprzedniej edycji)

Permutacją elementów danego zbioru nazywamy dowolne uporządkowanie elementów tego zbioru (czyli mamy tu bardziej do czynienia z ciągami, w których kolejność wystąpienia elementu ma znaczenie, niż ze zbiorami, gdzie taka kolejność jest nieistotna).

Przykładowo, mając zbiór $\{M, A, J, O, N, E, Z\}$, jego permutacją może być ten sam zbiór $\{M, A, J, O, N, E, Z\}$, albo zbiór $\{Z, N, A,\ J, O, M, E\}$, albo $\{A, M, N, E, Z, J, O\}$, albo np., układając litery w kolejności alfabetycznej: $\{A, E, J, M, N, O, Z\}$.

Można łatwo zauważyć, że jeśli elementy są różne, to zamiast przestawiać elementy, można przestawiać odpowiadające im ich indeksy. I tak w naszym przypadku wejściowy zbiór może przyjąć postać $\{1,2,3,4,5,6,7\}$, a kolejno wymienione powyżej jego permutacje to: $\{1,2,3,4,5,6,7\}$, $\{7,5,2,3,4,1,6\}$, $\{2,1,5,6,7,3,4\}$ i $\{2,6,3,1,5,4,7\}$.

Permutacje możemy również traktować jako funkcje, które przekształcają zbiór $\{1,2,\dots,n\}$ w siebie, przy czym konkretnej permutacji odpowiada konkretny porządek elementów. Przykładowo, w naszych przykładach $n = 7$, pierwsza z wymienionych permutacji $P_1(x) = x$, drugą P_2 opisać można następująco: $P_2(1) = 7,\ P_2(2) = 5,\ P_2(3) = 2,\ P_2(4) = 3,\ P_2(5) = 4,\ P_2(6) = 1,\ P_2(7) = 6$. Podobnie można by opisać pozostałe przedstawione permutacje. W związku z tym, w matematyce często daną permutację podaje się jako ciąg wartości tej permutacji dla kolejnych wartości argumentów. W powyższych przykładach mielibyśmy kolejno zapisy: $P_1 = (1\ 2\ 3\ 4\ 5\ 6\ 7)$, $P_2 = (7\ 5\ 2\ 3\ 4\ 1\ 6)$, $P_3 = (2\ 1\ 5\ 6\ 7\ 3\ 4)$, $P_4 = (2\ 6\ 3\ 1\ 5\ 4\ 7)$.

Od teraz zajmować się będziemy tylko permutacjami zbioru $\{1,2,\dots,n\}$, a daną permutację podawać będziemy w ostatniej postaci, czyli np. dla zbioru $n = 6$ moglibyśmy wybrać np. permutację $P = (5\ 2\ 1\ 3\ 4\ 6)$.

W matematyce przyjmuje się często inny zapis permutacji, wykorzystuje on tzw. rozkład na cykle. W ostatnim przypadku można symbolicznie zapisać:

$$\begin{array}{cccccc} (1 & 2 & 3 & 4 & 5 & 6) \\ \downarrow & \downarrow & \downarrow & \downarrow & \downarrow & \downarrow \\ (5 & 2 & 1 & 3 & 4 & 6) \end{array}$$

co oznacza, że 1 przechodzi na 5, zapisujemy to jak na razie: $(1\ 5$, dalej 5 przechodzi na 4, co wydłuża zapis do: $(1\ 5\ 4$, dalej 4 przechodzi na 3, co wydłuża zapis do: $(1\ 5\ 4\ 3$, dalej 3 przechodzi na 1, a liczba 1 pojawiła się na początku cyklu, więc cylk się zamyka, co zapisujemy: $(1\ 5\ 4\ 3)$. Nie zostały jednak wybrane wszystkie liczby ze zbioru $\{1,2,\dots,6\}$. Wybieramy pierwszą nieuwzględnioną jeszcze liczbę (jest to liczba 2), która rozpocznie kolejny cykl, mamy więc: $(1\ 5\ 4\ 3)(2$, ale ponieważ liczba 2 przechodzi na liczbę 2, która już wystąpiła w tym cyklu, to ten cykl też się zamyka. Otrzymujemy więc zapis: $(1\ 5\ 4\ 3)(2)$. Jedyną niewykorzystana jeszcze liczbą jest 6,

która też przechodzi na samą siebie, więc rozkład na cykle przyjmie teraz następującą postać $(1\ 5\ 4\ 3)(2)(6)$. Przyjmuje się również, że cykle długości jeden pomija się w końcowym zapisie, mamy więc ostatecznie: $(1\ 5\ 4\ 3)$.

Jeśli wzięlibyśmy permutację $P = (3\ 1\ 2\ 4\ 6\ 5)$, to:

$$\begin{matrix}(1 & 2 & 3 & 4 & 5 & 6)\\ \downarrow & \downarrow & \downarrow & \downarrow & \downarrow & \downarrow\\ (3 & 1 & 2 & 4 & 6 & 5)\end{matrix}$$

i widzimy, że 1 przechodzi na 3, 3 przechodzi na 2, 2 przechodzi na 1, co kończy pierwszy cykl. Najmniejszą niewykorzystana liczbą jest 4, która przechodzi na samą siebie, tworzący cykl długości 1. Teraz najmniejszą niewykorzystana liczbą jest 5, która przechodzi na 6, a 6 przechodzi na 5, co jest sygnałem zakończenia trzeciego (i ostatniego) cyklu. Mamy więc zapis $(1\ 3\ 2)(4)(5\ 6)$, i ostatecznie $(1\ 3\ 2)(5\ 6)$.

Dla permutacji $P = (4\ 6\ 5\ 1\ 3\ 2)$ mielibyśmy zapis postaci $(1\ 4)(2\ 6)(3\ 5)$, a dla permutacji $P = (1\ 5\ 2\ 4\ 3\ 6\ 8\ 7)$ mielibyśmy zapis $(1)(2\ 5\ 3)(4)(6)(7\ 8)$, a ostatecznie $(2\ 5\ 3)(7\ 8)$.

Permutacje są funkcjami, możemy więc je składać (tak jak możemy składać funkcje). Np. mając permutacje: $P_1 = (2\ 3\ 1\ 4)$, $P_2 = (3\ 1\ 4\ 2)$, ich złożenie będzie też permutacją: $P_1 \circ P_2 = P_1(P_2) = (2\ 3\ 1\ 4) \circ (3\ 1\ 4\ 2) = (1\ 2\ 4\ 3)$.

Napisz program, który dla zadanych permutacji wypisywał będzie ich złożenie. Zakładamy, że użytkownik może dane wprowadzać w dwóch postaciach: albo jako permutację, albo jej odpowiednik w postaci cykli, czyli odwołując się do ostatniego przykładu z rozkładu na cykle może to być $(1\ 5\ 2\ 4\ 3\ 6\ 8\ 7)$ lub $(2\ 5\ 3)(7\ 8)$. Można również „mieszać” te postacie i podawać argumenty w tych samych lub w różnych postaciach.

ZADANIE 4 – LICZBA ERDŐSA

Zadanie zaproponował: dr inż. Mariusz Pleszczyński, Wydział Matematyki Stosowanej, Politechnika Śląska (zadanie finałowe 2023)

Słynny węgierski matematyk Paul Erdős znany był, poza swoim geniuszem matematycznym, m.in. z tego, że publikował bardzo dużo prac z wieloma różnymi matematykami. W związku z tym, nieco żartobliwie, wprowadzona została liczba Erdősa klasyfikująca w pewien sposób publikujących matematyków.

Zakłada się, że Erdős ma liczbę Erdősa równą zero. Osoba, która razem z nim opublikowała artykuł ma liczbę Erdősa równą jeden. Osoba, która opublikowała pracę z kimś, kto opublikował pracę z Erdősem ma liczbę Erdősa równą 2, itd. Jeśli danej osobie można przyporządkować w ten sposób większą ilość liczb Erdősa, to wybieramy najmniejszą z możliwych takich liczb. Jeżeli danej osobie nie można w ten

sposób przypisać żadnej liczby Erdősa (np. wszystkie prace ma samodzielne), to przyporządkowujemy jej liczbę Erdős równą nieskończoność.

W pliku *publikacje.txt* znajduje się lista artykułów wg schematu:
Nazwisko_autora_1 I_1_1.I_1_2., Nazwisko_autora_2 I_2_1.I_2_2., …, ., Nazwisko_autora_n I_n_1.I_n_2.; Tytuł artykułu numer 1; inne dane o artykule pierwszym.
Nazwisko_autora_1 I_1_1.I_1_2., Nazwisko_autora_2 I_2_1.I_2_2., …, ., Nazwisko_autora_n I_n_1.I_n_2.; Tytuł artykułu numer 2; inne dane o artykule drugim.

Przykładowo lista taka mogłaby przyjąć postać:
Kowalski J., Nowak A.D.; Ciekawe rzeczy, neutralne rzeczy i rzeczy nieciekawe; Czasopismo Minut, Gliwice 2023.
Abacki L.E., Babacki T.R, Cabacki P., Dabakiewicz W.; O pochodzeniu nazwisk; Wydawnictwo Politechniki Wodzisławskiej, Wodzisław Śląski 2000.
Marek S.; Jak pisać samodzielne prace; wersja online, www.online.mareks.com.pl, widziane 23.02.12.

Autorów może być kilku, każdy z nich może mieć jedno lub dwa imiona (podane są tylko inicjały), artykułów może być wielu, definiowanie każdego z nich kończy znak nowej linii.

Zakładamy, że wśród wymienionych publikacji co najmniej jedna zawierała będzie nazwisko Erdősa (załóżmy, dla ułatwienia, że nazwisko to przyjmie następującą formę zapisu: *Erdos P.*).

Napisz program, który wczyta plik *publikacje.txt* i każdemu z występujących tam autorów poprawnie przyporządkuje odpowiadającą mu (tylko na podstawie tego pliku i informacji o tym, że Erdős ma liczbę Erdősa równą 0) liczbę Erdősa.

Uwaga: dla ułatwienia obok pliku *slownik.txt* podajemy, jako odpowiednie pliki .txt, przykładowe zestawy list artykułów i odpowiadających tym listom zestawów liczb Erdősa odpowiadających autorom występujących na tych listach.

ZADANIE 5 – METAMORFOZA

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

Jednym z zadań szaradziarskich jest *metamorfoza,* gdzie podane są dwa wyrazy o tej samej liczbie liter i takich, że żadna z liter słowa pierwszego nie jest literą drugiego słowa. Następnie należy wymieniać po jednej literze (wskazanej przez podanie jej pozycji) danego słowa (w pierwszym kroku jest to pierwsze słowo, w drugim – słowo powstałe przez wymianę w pierwszym słowie jednej litery, w trzecim – słowo powstałe przez wyminę niewymienionej do tej pory litery w słowie drugim, itd.) Przed wpisaniem kolejnego słowa litery można przestawiać miejscami (anagramować). Każda litera pierwszego słowa musi być zamieniona i każda litera drugiego z danych słów musi zostać wykorzystana.

Zadanie mogłoby przyjąć następującą formę (po lewej treść zadania, po prawej rozwiązanie z pomocniczym kolorowaniem liter):

S	E	T	A
			_
			_
			_
Z	G	O	N

→

S	E	T	A
A	T	O	S
T	O	G	A
G	O	N	T
Z	G	O	N

W pliku *słownik.txt* znajduje się słownik, w którym słowa (każde w nowej linii) posortowane są alfabetycznie.

Napisz program, który dla danego $2 < n \in \mathbb{N}$ poszukiwał będzie słów znajdujących się w słowniku, dla których istnieje „metamorfoza” jednego z tych słów w drugie. Oczywiście słowa „pośrednie” również muszą znajdować się w tym słowniku. Drugim argumentem programu jest liczba $m \in \mathbb{N}$, która odpowiada za ilość różnych par słów.

Program po wprowadzeniu n i m poszukiwał będzie odpowiednich m par słów, a jeżeli nie znajdzie m takich par, to wypisze $k < m$ takich par lub komunikat, że takich par słów nie ma. Program poda również (dla każdej z pary słów) słowa pośrednie i kolejne pozycje wymienianych liter (formę odpowiedzi pozostawiamy w gestii rozwiązującego).