Code Quest Przygotowanie do zawodów

Coroczne Międzynarodowe Zawody Programistyczne

Sobota, 30 kwietnia 2022 r.

Spis treści

Wstęp 2

Co można zrobić od razu 3

- Wybrać strategię dla waszego zespołu 3
- Przygotować środowisko pracy 3
- Rozwiązać problemy ćwiczeniowe 4
- Przygotować dane uwierzytelniające 4
- Przećwiczyć poprzednie problemy 4
- Używasz Visual Basic? Zapoznaj się z C# 4

Często zadawane pytania (FAQ) 5

Informacje matematyczne 7

- Zaokrąglanie 7
- Trygonometria 7

Tabela US ASCII 8

Terminologia 9

Problem A: Witaj, świecie! 10

- Wprowadzenie 10
- Opis problemu 10
- Przykładowe dane wejściowe 11
- Przykładowe dane wyjściowe 11
- Kod rozwiązania 11

Problem B: Pojazd nie całkiem autonomiczny 14

- Wzmianka o autorze 14
- Wprowadzenie 14
- Opis problemu 14
- Przykładowe dane wejściowe 14
- Przykładowe dane wyjściowe 15

Wstęp

Witamy w konkursie programistycznym Code Quest firmy Lockheed Martin! Cieszymy się na nasze spotkanie.

Zawody Code Quest są międzynarodowym, corocznym konkursem dla uczniów klas 9-12 (lub ich odpowiedników na całym świecie). Po ich założeniu w 2012 r. w zakładzie Fort Worth udało nam się rozpropagować Code Quest na całym świecie. Organizuje je 25 placówek w USA, Wielkiej Brytanii, Polsce, Australii, Nowej Zelandii i Singapurze. Nasza misja to zachęcanie uczniów do rozwijania swoich umiejętności i zainteresowań w zakresie języków programowania w ramach otwartego i pozostawiającego swobodę, ale i konkurencyjnego otoczenia. Dbamy również o dostępność materiałów dla nauczycieli, które wspomagają w podnoszeniu poziomu wykształcenia w dziedzinie programowania.

Waszym celem na czas zawodów jest ukończenie wyzwania poprzez zdobycie jak największej liczby punktów. Punkty zdobywa się pisząc programy komputerowe w celu rozwiązania szeregu problemów. Dostępne języki programowania to Java, Python, C# i C++. Czas zawodów to dwie i pół godziny. Wraz z całym waszym zespołem macie dokładnie tyle czasu na rozwiązanie jak największej liczby problemów.

Ponieważ pandemia COVID-19 wciąż stanowi istotną część naszego życia, tegoroczny konkurs ponownie odbędzie się w środowisku wirtualnym. Wasz zespół połączy się z konkursową konferencją Zoom, a tuż przed rozpoczęciem 2,5 godzinnym zawodów zostanie przesunięty do prywatnego pokoju ‘breakout room’ w ramach połączenia Zoom, co umożliwi omawianie pracy wraz z członkami zespołu bez osób trzecich. Ponieważ każdy z was będzie przy własnym komputerze, będziecie mogli jednocześnie pracować nad różnymi problemami! Dostarczymy wam listę 25 problemów, abyście mieli nad czym pracować podczas zawodów.

Pomimo obecnej sytuacji postaramy się poprowadzić je w sposób możliwie zbliżony do normalności. Jeśli podczas konkursu zetkniecie się z trudnościami, wolontariusze Lockheed Martin będą wam pomagać; możecie prosić o pomoc poprzez Zoom lub zadając pytania przez stronę konkursu. Mamy też dla was trochę zajęć jeszcze przed rozpoczęciem, gdy będziemy kończyć czynności organizacyjne, a także po zakończeniu, w trakcie zbierania wyników..

Pozostała część dokumentu objaśnia zasady działania konkursu i przedstawia problemy ćwiczeniowe, nad którymi możecie zacząć pracę od zaraz. W przypadku pytań w sprawie zawodów prosimy o kontakt waszego nauczyciela na adres codequest-poland.gr-sac@lmco.com, postaramy się odpowiedzieć jak najszybciej.

Powodzenia! Widzimy się 30 kwietnia!

Co można zrobić od razu

W celu jak najlepszego przygotowania się do zawodów powinniście zadbać o kilka spraw; przedstawiamy je poniżej.

Wybrać strategię dla waszego zespołu

Jak wspomniano powyżej, tegoroczne zawody będą wyglądać nieco inaczej. Każdy z was będzie siedział przy własnym komputerze zamiast zbierać się razem przy jednym w zatłoczonym pomieszczeniu. Musicie sami zdecydować, jaki model pracy zapewni wam największy sukces.

- Ustalcie swoje mocne i słabe strony - jeśli macie problemy z pewnym algorytmem lub jakąś koncepcją, jeden z członków waszego zespołu może być w stanie zaproponować wskazówki.
- Ustalcie, jak wybieracie problemy, z którymi się zmierzycie. Zaczynacie od łatwych i podejmujecie stopniowo coraz większe wyzwanie czy próbujecie wysforować się na czoło rankingu zaczynając od trudniejszego?
- Opracujcie strategię rozwiązywania problemów - czy każdy zajmuje się jednym z nich czy też razem rozmyślacie nad każdym? Co robić, gdy zatniecie się na jakimś problemie?

Przygotować środowisko pracy

Jeśli chodzicie na zajęcia z programowania lub macie komputer udostępniony przez szkołę, najprawdopodobniej macie już odpowiednie środowisko pracy, ale na wszelki wypadek przedstawiamy kilka narzędzi programistycznych zalecanych do problemów w zawodach Code Quest. Lista obejmuje programy instalowane na dysku, które zwykle są pewniejsze i mają więcej funkcji, ale również programy sieciowe, które lepiej nadają się do pracy zespołowej i mogą być niezbędne w przypadku pracy na komputerach szkolnych czy Chromebookach.

- **IDE instalowane na dysku:**
  - **Java/C++** - Eclipse IDE https://www.eclipse.org/downloads/
  - **Python** - Python IDLE (w większości plików instalacyjnych Python) https://www.python.org/downloads/
  - **C#** - Microsoft Visual Studio Community https://visualstudio.microsoft.com/vs/community/
- **IDE sieciowe**
  - **Replit** - https://repl.it/
  - **Codeanywhere -** https://codeanywhere.com/
- **Text editing** - Notepad++ https://notepad-plus-plus.org/downloads/
- **File comparisons** - Beyond Compare https://www.scootersoftware.com/

Kwestie prawne: Lockheed Martin nie jest powiązany z powyższymi stronami i ich właścicielami, firmami i produktami; nie poleca ich, nie autoryzuje i nie sponsoruje. Korzystacie z powyższych linków i wymienionych programów wyłącznie na własne ryzyko.

Rozwiązać problemy ćwiczeniowe

Gdy wszystko jest gotowe do pracy, przeczytajcie pozostałą część dokumentu, aby dowiedzieć się, jakie są zasady zawodów i czego oczekuje się od was i członków waszego zespołu. Na końcu dokumentu znajdują się problemy ćwiczeniowe, oznaczone literami A i B. Te problemy są już dostępne do rozwiązania na naszej stronie konkursowej. Rozwiązania problemu A podano w dokumencie, ale problem B musicie rozwiązać samodzielnie!

Przygotować dane uwierzytelniające

Wasz nauczyciel powinien otrzymać listę nazw użytkowników i haseł dla każdego członka waszego zespołu emailem. Sprawdźcie, czy znacie swoją nazwę użytkownika i hasło - każdy ma inne! Wasze hasło składa się z trzech małych liter, trzech wielkich liter, dwóch cyfr i jednego znaku specjalnego w losowej kolejności. Aby nie dopuścić do powstania wątpliwości, hasła nie będą zawierały:

- Mała litera „l” (jak w słowie „lampa”)
- Wielka litera „I” (jak w słowie „Ida”)
- Cyfra 0 (zero)

W przypadku problemów z logowaniem na stronie konkursowej w dniu zawodów można poprosić o pomoc wolontariuszy zespołu Lockheed Martin, którzy w razie potrzeby będą mieli możliwość zresetowania waszego hasła. W dniu zawodów zalecamy zalogować się możliwie jak najwcześniej, aby zyskać czas na usuwanie ewentualnych problemów technicznych i oczekiwanie na odpowiedzi na ewentualne pytania.

Przećwiczyć poprzednie problemy

Po przygotowaniu według powyższej listy jesteście gotowi do pracy! Jeśli chcecie zmierzyć się z innymi problemami Code Quest, możecie wejść na stronę naszej akademii pod adresem https://lmcodequestacademy.com. Możecie również pobierać problemy i ich rozwiązania z naszych konkursów z roku 2014 i wcześniejszych pod adresem https://www.lockheedmartin.com/en-us/who-we-are/communities/codequest.html.

Używasz Visual Basic? Zapoznaj się z C#

Code Quest wcześniej uwzględniał Visual Basic jako jeden z obsługiwanych języków w naszych konkursach, ze względu na jego użycie w programach nauczania, szczególnie w Wielkiej Brytanii i Australii. Jednakże, od czasu dodania tego języka, zauważyliśmy, że bardzo niewiele drużyn, nawet z tych krajów, używa go podczas konkursu. Ze względu na brak zainteresowania i problemy logistyczne związane z zapewnieniem wsparcia dla tego języka, Visual Basic nie będzie już akceptowany na zawodach Code Quest. Jeśli preferujesz język Visual Basic, przepraszamy za niedogodności, ale sugerujemy zapoznanie się z dokumentacją dla języka C#. C# jest podobnym językiem, który korzysta z frameworka .NET i Visual Basic, ale jest lepiej przystosowany do używania w programowaniu konkursowym.

Często zadawane pytania (FAQ)

Na czym polegają zawody?

W celu rozwiązania zadanego problemu wasz zespół ma za zadanie napisać program komputerowy, który odczytuje dane wejściowe ze standardowego kanału wejściowego i podaje dane wyjściowe do konsoli. W opisie każdego problemu podano format danych wejściowych i format, w jakim powinny zostać przedstawione dane wyjściowe. Po ukończeniu programu należy przesłać kod źródłowy programu na stronę internetową konkursu. Strona kompiluje i uruchamia kod, a uczestnicy otrzymują informację, czy odpowiedź jest prawidłowa czy nie.

Kto ocenia odpowiedzi?

W Lockheed Martin wyznaczono zespół pracowników odpowiedzialnych za ocenę rozwiązań konkursowych, ale większość tego procesu jest prowadzona automatycznie przez stronę konkursową. Strona kompiluje i uruchamia kod, a następnie porównuje dane wyjściowe z programu uczestnika do oficjalnych wyników. Jeśli są one jednakowe, wasz zespół otrzymuje punkty za podanie prawidłowej odpowiedzi.

Jak wygląda punktacja każdego problemu?

Każdy z nich ma przypisywaną wartość punktową w zależności od poziomu trudności. Gdy strona internetowa uruchamia wasz program, porównuje jego wyniki z oficjalnymi wynikami dla danego problemu. Jeśli są one jednakowe, wasz zespół otrzymuje punkty za rozwiązanie problemu. Za częściową zgodność punkty nie są przyznawane; dane wyjściowe z przesłanego programu muszą być dokładnie takie, jak oficjalne wyniki. Jeśli wasza odpowiedź została zweryfikowana negatywnie, a jesteście pewni, że jest prawidłowa, ponownie sprawdźcie format danych wyjściowych, a także, czy na końcu nie pojawiły się zbędne spacje lub inne niepotrzebne znaki.

Nie rozumiemy problemu. Kto może nam pomóc?

W przypadku trudności ze zrozumieniem problemu możecie przesyłać pytania do organizatorów za pośrednictwem strony internetowej konkursu. Nie możemy podpowiadać, jak dojść do rozwiązania, ale objaśniamy ewentualne niejasności. Jeśli zespół pracowników organizatora wykryje w trakcie zawodów błąd w jakimś problemie, możliwie jak najszybciej powiadomi o tym wszystkie uczestniczące w zawodach zespoły.

Nasz program działa z przykładowymi danymi wejściowymi/wyjściowymi, ale w ramach zawodów jest weryfikowany negatywnie! Dlaczego?

Pamiętajcie, że oficjalne dane wejściowe i wyjściowe służące do oceny waszych odpowiedzi są ZNACZNIE większe od przekazanych przykładowych danych. Te dane obejmują szerszy zakres badanych przypadków. Opis problemu zawsze wskazuje wartości graniczne danych wejściowych i wyjściowych, ale program musi być w stanie obsługiwać również przypadek wykraczający poza te granice. Wszystkie dane wejściowe i wyjściowe zostały gruntownie przetestowane przez nasz zespół ds. problemów i nie zawierają błędnych wartości.

Nie jesteśmy w stanie dojść do tego, dlaczego nasza odpowiedź jest nieprawidłowa. Co robimy źle?

Do najczęstszych błędów należą:

- Nieprawidłowe formatowanie - Warto dokładnie przyjrzeć się przykładowym danym wyjściowym podanym w problemie i sprawdzić, czy wasz program prezentuje wyniki w tym samym formacie.
- Nieprawidłowe zaokrąglenie - W następnym rozdziale opisano zaokrąglanie liczb dziesiętnych.
- Niewłaściwe liczby - 0 (a także 0,0, 0,00 itd.) NIE JEST liczbą ujemną. 0 może być dopuszczalną odpowiedzią, ale -0 już nie.
- Niepotrzebne znaki - Sprawdźcie, czy na końcu wierszy z wynikami nie ma dodatkowych spacji. Spacje na końcu wiersza nie są częścią danych wyjściowych w żadnym problemie.
- Format dziesiętny - We wszystkich wynikach liczbowych separatorem dziesiętnym jest kropka (.).

Jeśli te podpowiedzi są niewystarczające, możecie przesyłać pytania do organizatorów za pośrednictwem strony internetowej konkursu. Nie możemy podpowiadać, jak dojść do rozwiązania, ale zwykle możemy wyjaśnić, dlaczego dana odpowiedź jest zwracana jako nieprawidłowa.

Podczas przesyłania rozwiązania pojawia się błąd.

Przy przesyłaniu rozwiązań należy wybrać kod źródłowy dla swojego programu (w zależności od języka będą to pliki o rozszerzeniu .java, .cs, .cpp, lub .py). Należy pamiętać o przesłaniu wszystkich plików potrzebnych do skompilowania i uruchomienia programu. Należy również sprawdzić, czy nazwy plików nie zawierają spacji lub innych znaków spoza zakresu znaków alfanumerycznych (np. „Prob01.java” jest dopuszczalna, ale „Prob 01.java” i „Bob’sSolution.java” nie są).

Czy po zakończeniu zawodów mogę dostać rozwiązania problemów?

Tak! Poproście swojego opiekuna o przesłanie emaila na adres codequest-poland.gr-sac@lmco.com.

Jak ustala się kolejność w przypadku remisu?

Na koniec zawodów zespoły są szeregowane według liczby punktów uzyskanej z prawidłowych odpowiedzi na problemy. Jeśli w którejkolwiek kategorii konkursowej miejsca na podium są zajmowane przez zespoły z tą samą liczbą punktów, to kolejność ustala się według następujących kryteriów:

1. Mniej rozwiązanych problemów (co sugeruje, że w zespole rozwiązywano trudniejsze zadania)
2. Mniej nieprawidłowych odpowiedzi (co sugeruje, że popełniono mniej błędów)
3. Kolejność przesłania ostatniej prawidłowej odpowiedzi; im wcześniej, tym lepiej (co sugeruje, że pracowano szybciej)

Proszę pamiętać, że metody ustalania kolejności mogą nie być zawsze uwzględniane na wyświetlanej na żywo tabeli wyników na stronie internetowej zawodów. Ponadto, na 30 minut przed końcem zawodów tabela wyników zostaje zamrożona, dlatego należy przede wszystkim skupić się na wytężonej pracy!

Informacje matematyczne

Zaokrąglanie

W niektórych problemach zostaniecie poproszeni o zaokrąglanie liczb. We wszystkich stosuje się domyślnie zaokrąglanie „od połowy w górę” , chyba że podano inaczej w opisie problemu. W większości przypadków będzie to sposób, którego nauczyliście się w szkole, ale niektóre języki programowania stosują domyślnie inne metody zaokrąglania. **O ile nie jesteście pewni tego, jak wasz język programowania obsługuje zaokrąglanie, zalecamy napisanie własnego kodu do zaokrąglania liczb na podstawie informacji podanych w tym rozdziale.**

W metodzie zaokrąglania „od połowy w górę” liczby są zaokrąglane do najbliższej liczby całkowitej. Na przykład:

- 1.49 jest zaokrąglane w dół do 1.
- 1.51 jest zaokrąglane w górę do 2.

Od połowy w górę” oznacza, że gdy liczba jest dokładnie w połowie między dwoma liczbami całkowitymi, to jest zaokrąglana do liczby z większą wartością bezwzględną (czyli dalszą od 0). Na przykład:

- 1.5 jest zaokrąglane w górę do 2.
- -1.5 jest zaokrąglane w dół do -2.

Błędy zaokrąglania należą do najpowszechniejszych; jeśli w problemie występuje zaokrąglanie i strona internetowa odrzuca program jako podający nieprawidłowe wyniki, warto sprawdzić zaokrąglanie!

Trygonometria

Niektóre problemy mogą wymagać zastosowania funkcji trygonometrycznych, które pokrótce przedstawiono poniżej. Większość języków programowania ma wbudowane funkcje dla $\sin X$, $\cos X$ i $\tan X$; Należy uważnie przeczytać jego dokumentację. O ile nie podano inaczej w opisie problemu, *mocno zalecamy*, by używać wbudowanej w języku wartości dla pi (π), jeśli to koniecznie.

$$\sin X = \frac{A}{C} \quad \cos X = \frac{B}{C} \quad \tan X = \frac{A}{B} = \frac{\sin X}{\cos X}$$

$$X + Y = 90°$$

$$A^2 + B^2 = C^2$$

$$\frac{stopnie * \pi}{180} = radiany$$

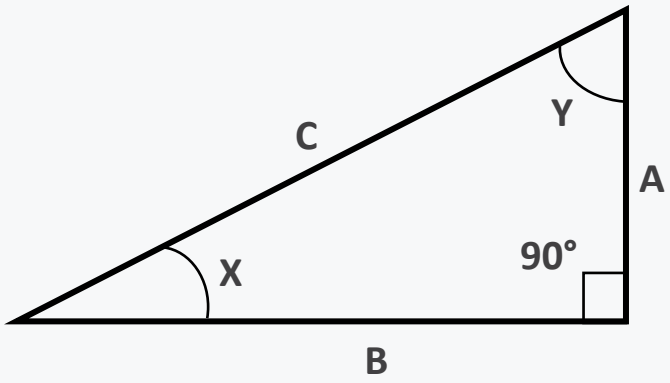

Tabela US ASCII

Dane wejściowe we wszystkich problemach Code Quest® korzystają ze znaków drukowalnych należących do tabeli znaków US-ASCII. Znaki niedrukowalne i sterujące nie są używane w problemach, chyba że wyraźnie zaznaczono to w opisie problemu. W niektórych przypadkach może pojawić się polecenie zamiany znaku na jego równoważnik liczbowy i/lub *vice versa*, zgodnie z poniższą tabelą.

Dwójkowy	Dziesiętny	Znak	Dwójkowy	Dziesiętny	Znak	Dwójkowy	Dziesiętny	Znak
0100000	32	(spacja)	1000000	64	@	1100000	96	`
0100001	33	!	1000001	65	A	1100001	97	a
0100010	34	"	1000010	66	B	1100010	98	b
0100011	35	#	1000011	67	C	1100011	99	c
0100100	36	$	1000100	68	D	1100100	100	d
0100101	37	%	1000101	69	E	1100101	101	e
0100110	38	&	1000110	70	F	1100110	102	f
0100111	39	'	1000111	71	G	1100111	103	g
0101000	40	(	1001000	72	H	1101000	104	h
0101001	41	)	1001001	73	I	1101001	105	i
0101010	42	*	1001010	74	J	1101010	106	j
0101011	43	+	1001011	75	K	1101011	107	k
0101100	44	,	1001100	76	L	1101100	108	l
0101101	45	-	1001101	77	M	1101101	109	m
0101110	46	.	1001110	78	N	1101110	110	n
0101111	47	/	1001111	79	O	1101111	111	o
0110000	48	0	1010000	80	P	1110000	112	p
0110001	49	1	1010001	81	Q	1110001	113	q
0110010	50	2	1010010	82	R	1110010	114	r
0110011	51	3	1010011	83	S	1110011	115	s
0110100	52	4	1010100	84	T	1110100	116	t
0110101	53	5	1010101	85	U	1110101	117	u
0110110	54	6	1010110	86	V	1110110	118	v
0110111	55	7	1010111	87	W	1110111	119	w
0111000	56	8	1011000	88	X	1111000	120	x
0111001	57	9	1011001	89	Y	1111001	121	y
0111010	58	:	1011010	90	Z	1111010	122	z
0111011	59	;	1011011	91	[	1111011	123	{
0111100	60	<	1011100	92	\	1111100	124	|
0111101	61	=	1011101	93	]	1111101	125	}
0111110	62	>	1011110	94	^	1111110	126	~
0111111	63	?	1011111	95	_			

Terminologia

W całym pakiecie opisujemy dane wejściowe i wyjściowe stosowane w waszych programach. W celu rozwiania wątpliwości pewne pojęcia są zawsze używane do określania różnych właściwości tych danych. Te pojęcia przedstawiono poniżej.

- **Liczba całkowita** to każda liczba bez ułamków zwykłych lub dziesiętnych. Liczbami całkowitymi są np. -5, 0, 5 i 123456789.
- **Liczba dziesiętna** to każda liczba, która nie jest liczbą całkowitą. Takie liczby mają separator dziesiętny i co najmniej jedną cyfrę na prawo od niego. -1.52, 0.0 i 3.14159 są liczbami dziesiętnymi.
- **Miejsca dziesiętne** oznaczają liczbę cyfr w liczbie dziesiętnej, która następuje po separatorze dziesiętnym. O ile nie podano inaczej w opisie problemu, liczby dziesiętne mogą zawierać dowolną liczbą miejsc dziesiętnych, równą co najmniej 1.
- **Liczba szesnastkowa** lub **łańcuch** stanowią szereg zawierający jeden znak lub więcej, w tym cyfry od 0 do 9 i/lub wielkie litery - A, B, C, D, E i/lub F. W tym konkursie małe litery nie są używane w wartościach szesnastkowych.
- **Liczby dodatnie** to liczby większe od 0. 1 to najmniejsza dodatnia liczba całkowita. 0.000000000001 to bardzo mała dodatnia liczba dziesiętna.
- **Liczby niedodatnie** to liczby, które nie są dodatnie; innymi słowy, wszystkie liczby mniejsze od 0 lub równe 0.
- **Liczby** ujemne to liczby mniejsze od 0. -1 to największa ujemna liczba całkowita. -0.000000000001 to bardzo duża ujemna liczba dziesiętna.
- **Liczby nieujemne** to liczby, które nie są ujemne; innymi słowy, wszystkie liczby większe od 0 lub równe 0.
- Włącznie oznacza, że przedział określony podanymi wartościami obejmuje również te podane wartości (czyli jest domknięty). Na przykład, przedział „od 1 do 3 włącznie” oznacza liczby 1, 2 i 3.
- **Wyłącznie** oznacza, że przedział określony podanymi wartościami nie obejmuje tych podanych wartości (czyli jest otwarty). Na przykład, przedział „od 0 do 4 wyłącznie” oznacza liczby 1, 2 i 3; 0 i 4 nie zawierają się w przedziale.
- **Formaty daty i godziny** są podawane literami, w miejsce których winny pojawić się liczby:
  - **HH** oznacza godziny, zapisywane dwoma cyframi (z ewentualnym zerem poprzedzającym). Opis problemu zawiera informację, czy należy używać formatu 12- czy 24-godzinnego.
  - **MM** oznacza minuty w przypadku czasu, a miesiące w przypadku dat. W obydwu przypadkach liczby są zapisywane dwoma cyframi (z ewentualnym zerem poprzedzającym; styczeń to 01).
  - **YY** lub YYYY oznacza rok, zapisywany dwoma lub czterema cyframi (z ewentualnym zerem poprzedzającym).
  - **DD** oznacza dzień miesiąca, zapisywany dwoma cyframi (z ewentualnym zerem poprzedzającym).

Problem A: Witaj, świecie!

Punkty: 0 (to problem ćwiczeniowy)

Wprowadzenie

Witamy w zawodach programistycznych Code Quest w roku 2022! Cieszymy się, że jesteście z nami. Życzymy wam powodzenia!

Przed rozpoczęciem zawodów chcemy, abyście upewnili się, że wasze połączenie sieciowe jest sprawne, a program oceniający dostępny. Nie trzeba do tego instalować ani uruchamiać żadnych programów na komputerze; wystarczy przeglądarka sieciowa, która powinna być już zainstalowana. Możecie połączyć się ze stroną konkursową otwierając URL podany w e-mailu, jaki wasz nauczyciel otrzymał przed zawodami, w którym podano też nazwy użytkowników i hasła.

Po połączeniu przejdźcie dalej; poniżej wyjaśniamy, jak napisać i przesłać rozwiązanie.

Opis problemu

Na początek przesyłacie przykładowy program, który ma potwierdzić, że wszystko działa poprawnie. Tutaj nie musicie przejmować się kodowaniem; podamy wam odpowiedź!

Choć dostarczymy wam przykładowe pliki wejściowe i wyjściowe do przetestowania waszych programów na komputerach, z których korzystacie, to jednak napisane programy muszą być w stanie odczytać dowolne dane wejściowe ze standardowego kanału wejściowego i podać je na standardowy kanał wyjściowy (konsolę). Takie IDE jak Eclipse i NetBeans można skonfigurować tak, by podawały treść otrzymanych plików bezpośrednio do danych wejściowych waszego programu. Jeśli uruchomicie program z poziomu wiersza poleceń (w środowisku Windows, Mac lub Linux), możecie podobnie przesłać treść pliku do standardowych danych wejściowych poniższymi poleceniami, zakładając, że pliki źródłowe i wejściowe są w katalogu bieżącym:

```
{polecenie kompilowania waszego programu}
{polecenie uruchomienia waszego programu} < {plik wejściowy}
```

Przykład w C++:

```
g++ -o Prob05.exe Prob05.cpp
Prob05.exe < Prob05.in.txt
```

Przykład w Java:

```
javac HelloWorld.java
java HelloWorld < HelloWorld.in.txt
```

tym przykładowym problemie wyświetlany wynik powinien odpowiadać przekazanym danym wejściowym. Przedstawiony kod źródłowy realizuje takie właśnie zadanie, ale możecie własny, jeśli jesteście pewni, że wszystko działa. Zalecamy również opracowanie rozwiązania dla problemu B.

W przypadku napotkania trudności teraz lub w trakcie zawodów prosimy o informowanie nas przez Zoom lub przesyłanie pytań za pośrednictwem strony konkursowej. Powodzenia!

Przykładowe dane wejściowe

Pierwszy wiersz danych wejściowych programu, **otrzymanych przez standardowy kanał wejściowy**, będzie zawierać dodatnią liczbę całkowitą oznaczającą liczbę przypadków testowych. Każdy przypadek testowy będzie zawierać pojedynczy wiersz tekstu, który ma zostać wyświetlony w standardowym kanale wyjściowym.

```
2
Welcome to Code Quest!
Good luck today!
```

Przykładowe dane wyjściowe

W każdym przypadku testowym program musi wyświetlić dane wejściowe bez zmian.

```
Welcome to Code Quest!
Good luck today!
```

Kod rozwiązania

Poniżej podano kod rozwiązania w każdym języku i szczegółowe informacje dotyczące pracy w każdym z języków. **Mocno zalecamy wykorzystanie tego kodu jako szablonu do rozwiązywania pozostałych problemów w trakcie zawodów**. Przesyłając rozwiązania załączajcie jedynie kod źródłowy (np. pliki .java, .cpp, .cs lub .py); nie załączajcie plików wykonywalnych (np. plików .exe) lub skompilowanego kodu (np. pliki .class).

Java

DOMJudge obsługuje Java 8. Korzystanie z oświadczeń „pakietowych” nie jest konieczne, ale nie spowoduje błędów kompilacji.

```java
import java.util.Scanner;

public class Prob00 {
  public static void main(String[] args) {
    try (Scanner input = new Scanner(System.in)){
      int testCases = Integer.parseInt(input.nextLine());

      for(int testcase = 0; testcase < testCases; testcase++) {
        System.out.println(input.nextLine());
      }
    }
  }
}
```

Python

DOMJudge obsługuje Python 3.

```
# Recommended imports for all problems
# Some problems may require more
import sys
import math
import string

cases = int(sys.stdin.readline().rstrip())
for caseNum in range(cases):
  print(sys.stdin.readline().rstrip())
```

C#

DOMJudge obsługuje .NET version 4.6 oraz C# version 6.0.

```
using System;

class CodeQuest {
  static void Main(string[] args) {
    int numTestCases = Convert.ToInt32(Console.ReadLine());

    for(int testCase = 0; testCase < numTestCases; testCase = testCase + 1){
      string text = Console.ReadLine();
      Console.WriteLine(text);
    }
  }
}
```

VB.NET

Visual Basic nie jest już wspierany podczas zawodów Code Quest, ze względu na brak zainteresowania ze strony drużyn oraz problemy logistyczne związane z zapewnieniem wsparcia dla tego języka. Jeśli Twoja drużyna używała Visual Basic w przeszłości, zalecamy zapoznanie się z dokumentacją dotyczącą C#; C# jest podobnym językiem, który również wykorzystuje framework .NET i jest lepiej przystosowany do programowania konkursowego.

C++

DOMJudge obsługuje wersję 7.3.0 kompilatora g++.

```
// Recommended includes for all problems. Some problems require additional
libraries.
#include <iostream>
#include <string>
#include <cmath>
#include <cstdlib>
using namespace std;

int main()
{
  int testCases;
  cin >> testCases;
```

```
  string dummy;
  getline(cin, dummy);

  for(int testcase = 0; testcase < testCases; testcase++){
    string text;
    getline(cin, text);
    cout << text << '\n';
  }
}
```

Problem B: Pojazd nie całkiem autonomiczny

Punkty: 0 (to problem ćwiczeniowy)

Autor: Chris Liu, California High School

Wzmianka o autorze

W przeciwieństwie do pozostałych problemów konkursowych tego problemu nie stworzył pracownik Lockheed Martin. Chris Liu był uczniem w California High School, który brał udział w zawodach Code Quest w Sunnyvale w Kaliforni w 2018 r. Od tamtej pory opracował dwa własne konkursy programistyczne, wzorowane na Code Quest. Chris jest również dyrektorem hackathonu SRC Hacks, który odbył się w lutym 2019 r. w San Ramon w Kaliforni, a w którym uczestniczyły setki uczniów szkół średnich. Chris opracował poniższy problem dla zawodów programistycznych w SRC Hacks. Przedstawiamy go za jego zezwoleniem, z drobnymi zmianami potrzebnymi do dopasowania go do naszego formatu.

Wprowadzenie

Dwayne i Johnson są razem z wami w grupowym projekcie w ramach zajęć inżynierskich dotyczących pojazdów autonomicznych. Niestety nikt z was nie uważał podczas zajęć i zapomnieliście o projekcie. Przypomnieliście sobie o nim dzień przed terminem oddania! Wiedząc, że zabraknie czasu na stworzenie porządnego algorytmu jazdy autonomicznej w zaledwie kilka godzin postanowiliście zasymulować pojazd autonomiczny, używając jedynie kilku czujników i instrukcji warunkowych.

Opis problemu

Waszym zadaniem jest opracowanie systemu unikania przeszkód, który wyświetla właściwe instrukcje jazdy w zależności od odległości od przeszkód i prędkości samochodu.

Przykładowe dane wejściowe

Pierwszy wiersz danych wejściowych programu, **otrzymanych przez standardowy kanał wejściowy**, będzie zawierać dodatnią liczbę całkowitą oznaczającą liczbę przypadków testowych. Każdy przypadek testowy będzie zawierać pojedynczy wiersz tekstu, zawierający dwie liczby oddzielone dwukropkiem:

- Wartość dziesiętną **V**, od 0 do 200 włącznie, wskazującą bieżącą prędkość pojazdu (w metrach na sekundę).
- Wartość dziesiętną **X**, od 1 do 400 włącznie, wskazującą odległość od przeszkody znajdującej się przed samochodem (w metrach).

Obydwie wartości zaokrągla się do dwóch miejsc dziesiętnych.

```
5
23.15:98.34
2.40:17.33
6.79:5.01
0.00:1.53
113.56:113.56
```

Przykładowe dane wyjściowe

W każdym przypadku testowym program musi wyświetlić instrukcje jazdy dla samochodu, według poniższych kryteriów:

- Jeśli, przy zachowaniu aktualnej prędkości, samochód zderzy się z przeszkodą najpóźniej za jedną sekundę, to program powinien wyświetlić SWERVE (skręć).
- Jeśli, przy zachowaniu aktualnej prędkości, samochód zderzy się z przeszkodą najpóźniej za pięć sekund, to program powinien wyświetlić BRAKE (hamuj).
- W przeciwnym razie program powinien wyświetlać SAFE (bezpiecznie).

```
BRAKE
SAFE
SWERVE
SAFE
SWERVE
```