Instrukcja przed etapem wojewódzkim konkursu zDolny Ślązak z informatyki

Na etapie wojewódzkim zawodnicy, pracując w warunkach kontrolowanej samodzielności na komputerach dostarczonych przez organizatorów, rozwiązują zadania programistyczne, implementując programy komputerowe w języku C++ lub Python i wysyłając je przez sieć do systemu konkursowego Solve 4 (`https://solve.edu.pl`).

Treści zadań konkursowych będą dostępne elektronicznie w systemie konkursowym.

1 Maszyna zawodnicza

Zawodnicy mają dostęp do komputera pracującego pod kontrolą systemu Linux ze środowiskiem graficznym i podstawowym pakietem narzędzi programistycznych dla języka C++ oraz Python.

W szczególności zawodnicy mają dostęp do:

- kompilatora `g++` oraz interpretera `python3`, które są stosowane na maszynach sprawdzających,
- terminala (powłoki `bash`),
- edytorów tekstu z kolorowaniem składni C++ i Pythona (np. `gedit`, `kate` lub `mousepad`),
- środowisk programistycznych (np. Visual Studio Code lub VS-Codium),
- narzędzi do debugowania (np. `gdb`),
- przeglądarki internetowej (np. Chrome lub Firefox) umożliwiającej dostęp do systemu konkursowego oraz dokumentacji języków i ich bibliotek standardowych,
- kalkulatora systemowego.

Zwracamy uwagę, że nie jest gwarantowane, że na maszynach zawodniczych będą zainstalowane dodatkowe popularne wtyczki do środowisk programistycznych (np. Github Copilot lub wtyczki ułatwiające pracę w środowisku z użyciem języków C++/Python).

2 Wysyłanie rozwiązań i użycie sieci

Programy powinny składać się z jednego pliku źródłowego, wczytywać dane ze standardowego wejścia (np. `cin` w C++ lub `input` w Pythonie) i wypisywać odpowiedzi na standardowe wyjście (np. `cout` w C++ lub `print` w Pythonie). Podczas zawodów dostęp do sieci jest ograniczony jedynie do systemu konkursowego oraz dokumentacji języków C++ i Pythona i ich bibliotek standardowych. Organizatorzy monitorują użycie sieci i w przypadku stwierdzenia celowych naruszeń (na przykład próby dostępu do internetu, wcześniej przygotowanych plików lub komunikacji z innymi zawodnikami lub narzędziami sztucznej inteligencji) mogą podjąć decyzję o dyskwalifikacji zawodnika.

3 Ocena rozwiązań

Programy zawodników są oceniane automatycznie przez system konkursowy, a zawodnicy informowani są o wyniku oceny natychmiast, gdy system sprawdzi zgłoszenie (zazwyczaj w ciągu kilku–kilkunastu sekund od wysłania rozwiązania). Zawodnicy mogą zgłaszać swoje rozwiązania do każdego zadania wielokrotnie. Każde zadanie oceniane jest w skali 0–100. Wynikiem zawodnika za zadanie jest najwyższy wynik punktowy uzyskany za zgłoszenie do danego zadania. Wynikiem zawodnika za zawody finałowe jest suma punktów uzyskanych na wszystkich zadaniach.

System konkursowy obsługiwany jest przez kilka maszyn sprawdzających, zdolnych do równoległego sprawdzania prac wielu zawodników. Mimo to, zachodzi możliwość (szczególnie w ostatnich minutach konkursu, gdy wielu zawodników wysyła swoje zgłoszenia), że czas sprawdzania zgłoszeń będzie dłuższy niż zwykle i na wynik zgłoszenia trzeba będzie poczekać. Nie może być to powodem do reklamacji. Uwaga: Zawodnik może w tym czasie robić coś innego.

3.1 Kompilacja i poprawność składniowa programu

Ocena programu zawodnika odbywa się zawsze na maszynie sprawdzającej, do której zawodnik nie ma bezpośredniego dostępu i rozpoczyna się od skompilowania kodu źródłowego (w przypadku C++) lub sprawdzenia poprawności składniowej programu (w przypadku Pythona). Jeżeli ten etap zakończy się błędem, zawodnik jest o tym informowany (zgłoszenie otrzymuje status `compilation error`) i ma dostęp do loga z kompilacji/sprawdzenia składni. Zgłoszenia z tym statusem otrzymują zero punktów.

Wszystkie maszyny sprawdzające mają jednakową konfigurację, wszystkie mają procesor Intel Core i5-6200U @2.30 GHz i pracują pod kontrolą systemu Linux. Programy w C++ kompilowane są z użyciem kompilatora `g++` z pakietu GCC w wersji 10.2. Do kompilacji stosowane są następujące flagi: `-O2 -Wall -Wshadow -Wextra -x c++ -lm -std=c++17`.

System konkursowy umożliwia również wybór standardu C++20 (flagi `-std=c++20`). Zwracamy uwagę, że wsparcie dla tego standardu w kompilatorze jest tylko częściowe i eksperymentalne, a zawodnik, który zdecyduje się wysyłać zgłoszenia z użyciem tego standardu czyni to na własne ryzyko. Programy w Pythonie są uruchamiane za pomocą interpretera CPython w wersji 3.9.16. Sprawdzanie pod kątem składni następuje z użyciem polecenia `py_compile`, a uruchamianie z użyciem polecenia `python3`. Dodatkowe biblioteki ponad bibliotekę standardową (w szczególności `boost` dla C++ lub `numpy`, `pandas` dla Pythona) nie są dostępne i próba skorzystania z nich spowoduje uzyskanie statusu `compilation error` (w C++) lub `runtime error` (w Pythonie).

Czas kompilacji/sprawdzenia składni nie może przekroczyć 12 sekund, a pamięć użyta do tego nie może przekroczyć 512 MB. Maksymalny rozmiar skompilowanego programu (w przypadku C++) nie może przekroczyć 32 MB. Rozmiar kodu źródłowego wysyłanego do systemu jest ograniczony do 100 kB.

3.2 Uruchomienie programu

Programy, które prawidłowo się kompilują/są poprawne składniowo podlegają uruchomieniu na zestawie wcześniej przygotowanych przez organizatorów testów. Każdy test to dane wejściowe (lub określony sposób interakcji z programem zawodnika), któremu odpowiada określone oczekiwane wyjście programu zawodnika (lub określony sposób interakcji z programem jurorskim). W treści każdego zadania znajdują się precyzyjne informacje o specyfikacji wejścia i wyjścia. Można założyć, że wszystkie testy spełniają tę specyfikację i nie ma potrzeby sprawdzania poprawności tych danych.

W każdym zadaniu podany jest również zestaw jednego lub więcej testów przykładowych. Obrazują one format wczytywania i wypisywania odpowiedzi oraz stanowią dodatkowe wyjaśnienie treści zadania. Zwracamy uwagę, że testy przykładowe, choć są w zestawie testów używanych do oceny rozwiązań, są warte zero punktów (niezależnie od tego, czy program zawodnika zalicza te testy czy nie). Do wyznaczenia wyniku punktowego zgłoszenia używany jest inny (niejawny podczas trwania zawodów) zbiór testów.

Jeżeli w trakcie wykonania programu zawodnika nastąpi błąd (np. dzielenie przez zero lub niedozwolony dostęp do pamięci), system konkursowy zgłasza to statusem `runtime error`. Może się tak również wydarzyć, jeżeli program kończy swoje działanie z kodem wyjścia innym niż 0. Uzyskanie tego statusu oznacza uzyskanie zero punktów za dany test.

Programy zawodników uruchamiane są w ograniczonym środowisku uruchomieniowym, z ograniczonymi zasobami (dostęp do plików i sieci). Niedozwolone jest w szczególności otwieranie plików tymczasowych lub korzystanie z wielowątkowości. System konkursowy wykrywa niektóre próby ingerowania w te zabezpieczenia i w takim przypadku przerywa wykonanie programu zgłaszając status `rule violation`. Uzyskanie tego statusu oznacza uzyskanie zero punktów za dany test.

Organizatorzy sprawdzają zgłoszenia zawodników i w przypadku stwierdzenia, że naruszenie zostało dokonane celowo, mogą podjąć decyzję o dyskwalifikacji zawodnika. Taka decyzja może być też podjęta w przypadku zawodników, którzy celowo próbują naruszyć bezpieczeństwo i stabilność systemu konkursowego (na przykład celowo wysyłając zbyt wiele zapytań do systemu lub zgłaszając programy do sprawdzania ponad miarę).

3.3 Limity czasu i pamięci

Dla każdego testu przypisany jest osobny limit czasu oraz limit pamięci, którego program zawodnika nie może przekroczyć (w przeciwnym razie system konkursowy informuje o tym jednym ze statusów: `time limit exceeded`, `memory limit exceeded` lub `runtime error`). Uzyskanie tego statusu oznacza uzyskanie zero punktów za dany test.

Limity czasu i pamięci podane są w treści zadań. Limit pamięci jest jednakowy dla wszystkich testów. Limit czasu podany w treści zadania oznacza maksymalny limit czasu, który jest stosowany dla największych testów. Możliwe, że na niektórych testach limit czasu będzie mniejszy. Organizatorzy pozostawiają sobie pole do decyzji, które testy są maksymalne.

Wszystkie limity dla C++ i Pythona są jednakowe. Zachodzi ryzyko, że ze względu na specyfikę narzędzi, programy napisane w Pythonie będą działały (wielokrotnie) wolniej niż ich analogiczne odpowiedniki napisane w C++, co w niektórych przypadkach może mieć wpływ na uzyskaną punktację. Intencją organizatorów jest jednak, żeby kalibracja limitów była taka, żeby maksymalną punktację można było uzyskać implementując rozwiązanie do każdego zadania zarówno w C++ jak i w Pythonie.

3.4 Poprawność wyjścia

Jeżeli program zakończy się bez błędu uruchomienia i nie przekroczy limitu czasu oraz pamięci, odbywa się sprawdzenie wygenerowanego wyjścia (lub interakcji z programem jurorskim). Jeżeli wygenerowana odpowiedź (lub interakcja z programem jurorskim) jest błędna, system konkursowy informuje o tym statusem `wrong answer`. Uzyskanie tego statusu oznacza uzyskanie zero punktów za dany test. Obok tego statusu może być również wyświetlona dodatkowa informacja o dokładnych przyczynach błędu (np. *Wczytano 7, oczekiwano 5.*). Próby nadużywania tej informacji (na przykład poprzez wysyłanie programów, których celem jest wyprodukowanie stosownego komunikatu, żeby ujawnić sobie zbiór danych wejściowych i odpowiedzi do nich) będą monitorowane przez organizatorów i mogą być karane unieważnianiem zgłoszeń, a w szczególnie rażących przypadkach dyskwalifikacją zawodnika.

3.5 Częściowa ocena rozwiązań

W niektórych zadaniach możliwe jest, że odpowiedź programu zawodnika będzie jedynie częściowo poprawna (w treści zadania mogą być podane warunki częściowej oceny). W takiej sytuacji (oraz w sytuacji, gdy odpowiedź będzie w pełni poprawna) system konkursowy informuje o tym statusem `accepted`. Obok tego statusu może być wyświetlona dodatkowa informacja o przyznanych punktach za dany test lub informacja o częściowej/pełnej poprawności odpowiedzi.

Intencją organizatorów jest to, że ocena programu zależy w pierwszej kolejności od jego poprawności (algorytmicznej oraz implementacyjnej), a w drugiej kolejności od jego wydajności. W związku z tym rozwiązania algorytmicznie niepoprawne lub zawierające błędy implementacyjne mogą być ocenione bardzo nisko (nawet na zero punktów), zaś rozwiązania poprawne, ale mało efektywne mogą uzyskiwać częściową punktację, zależnie od stopnia efektywności. W niektórych zadaniach możliwe jest, że zostanie podana pewna informacja na temat zbioru testów (na przykład, że w testach wartych pewną liczbę punktów zachodzą dodatkowe warunki ułatwiające rozwiązanie zadania: na przykład dane są mniejszego rozmiaru lub nie zawierają pewnych złośliwych lub trudnych przypadków). Można założyć, że zbiór danych testowych spełnia specyfikację warunków częściowej oceny i korzystać z tej informacji przy pisaniu programów.

3.6 Grupowanie testów

Przy ustalaniu wyniku punktowego zgłoszenia stosowane jest grupowanie testów: zbiór jednego lub więcej testów może stanowić grupę, a wynikiem zawodnika za grupę testów jest minimalny wynik punktowy z testów wchodzących w skład tej grupy. Celem grupowania testów jest sprawiedliwa ocena rozwiązań, które próbują zgadywać odpowiedź (na przykład zawsze wypisują przewidywalną odpowiedź typu `NIE`, zero lub losowa liczba). Takie rozwiązania, chociaż mogą prawidłowo zadziałać na niektórych testach, mogą zostać ocenione bardzo nisko (nawet na zero punktów). Wynikiem programu zawodnika jest suma punktów uzyskanych z wszystkich grup testów przypisanych do tego zadania.

4 Sytuacje sporne

4.1 Pytania dotyczące treści zadań

System konkursowy umożliwia umieszczanie publicznych ogłoszeń dla zawodników. Zawodnikom rekomenduje się regularne sprawdzanie informacji tam zawartych.

System konkursowy umożliwia również zadawanie pytań dotyczących treści zadań. Należy mieć na uwadze, że odpowiedzi udzielają organizatorzy i że na odpowiedź trzeba poczekać (warto w tym czasie robić coś innego). W niektórych przypadkach organizatorzy mogą nie udzielić odpowiedzi na pytanie (na przykład, gdy odpowiedź na pytanie jest zawarta w treści zadania lub gdy udzielenie odpowiedzi mogłoby stanowić podpowiedź w rozwiązaniu zadania).

4.2 Problemy techniczne

System konkursowy zapamiętuje wszystkie zgłoszone rozwiązania i umożliwia ich pobranie. Można wykorzystać tę funkcjonalność do tworzenia kopii zapasowych swojej pracy.

W razie problemów technicznych z komputerem lub dostępem do sieci, zawodnik powinien natychmiast zgłosić sytuację osobom pilnującym obecnym na sali, które zanotują dane niezbędne do ustalenia czasu awarii oraz podejmą próbę naprawienia problemu. Jeżeli okaże się, że rozwiązanie problemu nie jest możliwe w rozsądnym czasie, zawodnik otrzyma nowe stanowisko i będzie mógł przenieść tam swoje rozwiązania z kopii zapasowej utworzonej w systemie konkursowym. Jeżeli problem nie wynikał z winy zawodnika, organizatorzy mogą podjąć decyzję o przedłużeniu czasu zawodnikowi proporcjonalnie do czasu trwania awarii. Zwracamy uwagę, że zawodnikowi nie przysługuje dodatkowe przedłużenie czasu na pliki, których nie uda się odzyskać ze względu na brak kopii zapasowej utworzonej w systemie konkursowym.

4.3 Błędy i odwołania

Organizatorzy dołożą wszelkich starań, żeby system konkursowy, treści zadań, odpowiadające im dane testowe oraz inne narzędzia stosowane do oceny rozwiązań były poprawne, jednak w przypadku wykrycia poważnych błędów zatrzegają sobie możliwość do ich poprawiania, również w trakcie konkursu lub po jego zakończeniu, zawsze na korzyść dotkniętych błędami zawodników.

Po konkursie, w systemie konkursowym udostępnione zostaną wszystkie materiały użyte do oceny rozwiązań (w szczególności dane testowe oraz odpowiedzi do nich). Możliwe będzie również wysyłanie rozwiązań po konkursie. System konkursowy sprawdzi zgłoszenia wysłane po konkursie, jednak wynik tej oceny nie będzie wiążący dla wyników ostatecznych konkursu.

Zawodnicy (i ich prawni opiekunowie) będą mogli składać odwołania od wyników oceny zgłoszeń wysłanych w trakcie konkursu zgodnie z regulaminem ogólnym. Zwracamy uwagę, że dobór testów oraz limitów (zgodnych ze specyfikacją zadań) nie podlega odwołaniu.

5 Runda próbna

Na stronie `https://solve.edu.pl/contest/root-zdolnyslazak-2026-practice` znajduje się runda próbna zawierająca kilka przykładowych zadań, obrazujących format planowanych zawodów. Zwracamy uwagę, że liczba zadań oraz poziom ich trudności na zawodach właściwych może być inna niż na rundzie próbnej. Celem rundy próbnej jest zapoznać zawodników z systemem konkursowym oraz formatem zadań przed zawodami właściwymi. Po zarejestrowaniu konta w systemie Solve 4 możliwe jest wysyłanie rozwiązań i przeglądanie wyników zgłoszeń w taki sam sposób, jak będzie można to robić na zawodach właściwych.

Na następnych stronach przedstawiamy i omawiamy pierwsze przykładowe zadanie z rundy próbnej, zwracając jednocześnie uwagę, że w rundzie próbnej znajduje się więcej zadań. Zachęcamy do spróbowania.

5.1 Przykładowe zadanie

Jędrzej poszedł do sklepu kupić sobie wymarzony napis złożony z liter `a` oraz `b`. W sklepie każda litera `a` kosztuje bajtalara, zaś każda litera `b` kosztuje dwa bajtalary. Możliwe jest jednak zakupienie pakietu czterech liter `a` za trzy bajtalary lub pakietu pięciu liter `b` za osiem bajtalarów, ale tylko wtedy, gdy te litery stanowią spójny fragment kupowanego napisu.

Jędrzej zastanawia się za ile może zakupić swój wymarzony napis. Pomóż mu! Napisz program, który wczyta wymarzony napis Jędrzeja, wyznaczy jego koszt i wypisze wynik na standardowe wyjście.

Wejście W pierwszym (jedynym) wierszu wejścia znajduje się niepusty ciąg małych liter `a` lub `b`, bez żadnych odstępów. Jest to wymarzony napis Jędrzeja.

Wyjście W pierwszym (jedynym) wierszu wyjścia powinna się znaleźć jedna liczba całkowita – minimalny koszt napisu z wejścia według zasad panujących w sklepie.

Ograniczenia Długość napisu nie przekracza 1 000 000 znaków.

Gwarantowane jest, że w testach wartych łącznie 50% punktów w napisie na wejściu nie występują cztery lub więcej liter `a` obok siebie ani pięć lub więcej liter `b` obok siebie.

Przykład

Wejście:

```
aabaaaaabba
```

Wyjście:

```
13
```

Wyjaśnienie:
Możliwe jest zakupienie jednego pakietu trzech liter `a`. Pozostałe znaki trzeba zakupić indywidualnie. Koszt wyniesie więc: $1 + 1 + 2 + 3 + 1 + 2 + 2 + 1 = 13$ bajtalarów.

5.2 Rozwiązania przykładowego zadania

Wzorcowe rozwiązanie w C++:

```cpp
#include <bits/stdc++.h>
using namespace std;

int main() {
  ios_base::sync_with_stdio(false); cin.tie(0);

  string napis;
  cin >> napis;

  int koszt = 0, ostatnie_a = 0, ostatnie_b = 0;
  for (char znak : napis) {
    if (znak == 'a') {
      koszt++;
      ostatnie_a++;
      ostatnie_b = 0;
      if (ostatnie_a == 4) {
        koszt--;
        ostatnie_a = 0;
      }
    }
    if (znak == 'b') {
      koszt += 2;
      ostatnie_b++;
      ostatnie_a = 0;
      if (ostatnie_b == 5) {
        koszt -= 2;
        ostatnie_b = 0;
      }
    }
  }

  cout << koszt << "\n";

  return 0;
}
```

Wzorcowe rozwiązanie w Pythonie:

```python
napis = input()

koszt, ostatnie_a, ostatnie_b = 0, 0, 0
for znak in napis:
  if znak == 'a':
    koszt += 1
    ostatnie_a += 1
    ostatnie_b = 0
    if ostatnie_a == 4:
      koszt -= 1
      ostatnie_a = 0
  if znak == 'b':
    koszt += 2
    ostatnie_b += 1
    ostatnie_a = 0
    if ostatnie_b == 5:
      koszt -= 2
      ostatnie_b = 0

print(koszt)
```

Rozwiązanie za 50 punktów w C++:

```cpp
#include <bits/stdc++.h>
using namespace std;

int main() {
  ios_base::sync_with_stdio(false); cin.tie(0);

  string napis;
  cin >> napis;

  int wynik = 0;
  for (char znak : napis) {
    if (znak == 'a') wynik++;
    if (znak == 'b') wynik += 2;
  }

  cout << wynik << "\n";

  return 0;
}
```

Rozwiązanie za 50 punktów w Pythonie:

```
napis = input()

wynik = 0
for znak in napis:
  if znak == 'a': wynik += 1
  if znak == 'b': wynik += 2

print(wynik)
```