Zadanie zaliczeniowe 3: Problem Collatza

Wprowadzenie

Problem Collatza, zwany też czasami problemem 3x + 1 to słynny otwarty problem z dziedziny informatyki. W skrócie, chodzi o odpowiedź na pytanie czy następująca pętla się zatrzyma dla każdego n >= 2:

```
while (n != 1)
    if (n % 2 == 0) { n = n / 2; }
    else { n = 3 * n + 1; }
```

Pewnego dnia spacerując ulicą Dobrą, zauważył go na ścianie Biblioteki Uniwersytetu Warszawskiego znany milioner - pan Bajtazar. Postanowił raz na zawsze rozwiązać tę zagadkę. W tym celu zorganizował konkurs, którego celem było wyłonienie zespołu, który będzie potrafił rozwiązywać instancję problemu jak najszybciej.

Do konkursu zgłosiło się aż 13 zespołów, a każdy z nich postanowił podejść do problemu trochę inaczej. Celem zadania jest zaimplementowanie konkursu i porównanie wyników zespołów.

Opis zadania

Zadaniem studenta jest uzupełnienie kodu dołączonego w pliku collatz.zip, a następnie przygotowanie raportu porównującego wyniki zespołów. W pliku main.cpp znajduje się kod testujący każdy z zespołów w szeregu eksperymentów. Aby testy zakończyły się z sukcesem należy uzupełnić implementację zespołów w pliku teams.cpp, uwzględniając zawartość pliku teams.hpp. Każdy zespół powinien wywoływać w odpowiedni dla niej sposób funkcję calcCollatz(...) z pliku collatz.hpp. Zespoły powinny być zaimplementowane w następujący sposób:

- *TeamSolo* to jednoosobowy zespół wykorzystujący jeden proces i jeden wątek. Jego implementacji nie trzeba modyfikować. Należy go użyć jako punktu odniesienia dla pozostałych zespołów.
- *TeamNewThreads* powinien tworzyć nowy wątek dla każdego wywołania calcCollatz, jednak nie więcej niż getSize() wątków jednocześnie.
- *TeamConstThreads* powinien utworzyć getSize() wątków, a każdy z wątków powinien wykonać podobną, zadaną z góry ilość pracy.
- *TeamPool* powinien użyć dołączonej puli wątków cxxpool::thread_pool. Dokumentacja puli znajduje się na githubie (https://github.com/bloomen/cxxpool)
- *TeamNewProcesses* powinien tworzyć nowy proces dla każdego wywołania calcCollatz, jednak nie więcej niż getSize() procesów jednocześnie.
- *TeamConstProcesses* powinien tworzyć getSize() procesów, a każdy z procesów powinien wykonać podobną, zadaną z góry ilość pracy.
- *TeamAsync* powinien użyć mechanizmu std::async. W przeciwieństwie do pozostałych drużyn nie limituje on zasobów.

Każdy ze współbieżnych zespołów ma swojego klona - odpowiadający mu zespół X (np. TeamPoolX, TeamAsyncX). Zespoły X nie muszą używać funkcji calcCollatz z pliku collatz.hpp, mogą użyć swojej własnej implementacji. W szczególności oczekuje się, że zespoły działające w obrębie jednego procesu wykorzystają dzieloną strukturę SharedResults (plik sharedresults.hpp). SharedResults powinna przechowywać wybrane wyniki częściowe, aby (przynajmniej w teorii) przyspieszyć obliczenia. Ustalenie interfejsu i implementacja struktury SharedResults to część zadania. W przypadku zespołów tworzących procesów wymagana jest komunikacja między procesami. Można np. użyć mechanizmów takich jak pamięć dzielona poznanych na laboratorium w języku C.

Podsumowanie wyników konkursu powinno zostać opisane w raporcie, który przedstawia czas działania każdego zespołu w różnych scenariuszach. Czas działania powinien być przedstawiony na czytelnych

wykresach, na podstawie danych zbieranych przez obiekty klasy DefaultTimer. Dla każdego konkursu uzasadnij dlaczego konkretne drużyny były szybsze, a inne wolniejsze. Rozwiązania wielowątkowe powinny w uzasadnionych przypadkach być istotnie szybsze, jednak nie należy spodziewać się liniowego przyspieszenia w każdym z przypadków. Opisując wyniki uwzględnij rezultaty "CalcCollatzSoloTimer" dla różnych konkursów. W uzasadnieniu możesz również uwzględnić swoje własne timery, jeśli je dodasz. Eksperymenty należy przeprowadzić w dwóch różnych środowiskach np. komputer stacjonarny / laptop i maszyna students. Zadbaj o opis środowiska w raporcie, w szczególności model procesora wraz z liczbą rdzeni. Zwróć uwagę na różnice w wynikach które pojawią się przy wielokrotnym uruchamianiu programu. Idealny raport powinien zmieścić się na dwóch stronach a4.