Zadanie

Napisz oprogramowanie w C#, Python bądź Go, które monitoruje ilość czasu spędzonego w budynku przez osobę w ciągu dnia. Plik wykonywalny programu nie powinien wymagać żadnych parametrów podawanych w wierszu poleceń. Oprogramowanie jako źródło danych ma pobierać domyślnie plik 'input.csv'. Plik domyślnie zawiera trzy kolumny 'Date', 'Event', 'Gate'.

Kolumna

- 'Date' zawiera datę w formacie: yyyy-mm-dd hh:mm:ss
- 'Event' zawiera informację o rodzaju zdarzenia czy czujnik znajdował się w biurze, czy poza biurem
- 'Gate' zawiera informację na temat identyfikatora drzwi

Wynikiem działania programu ma być plik wyjściowy o nazwie 'result' (bez rozszerzenia).

Plik 'result' ma zawierać informacje jak

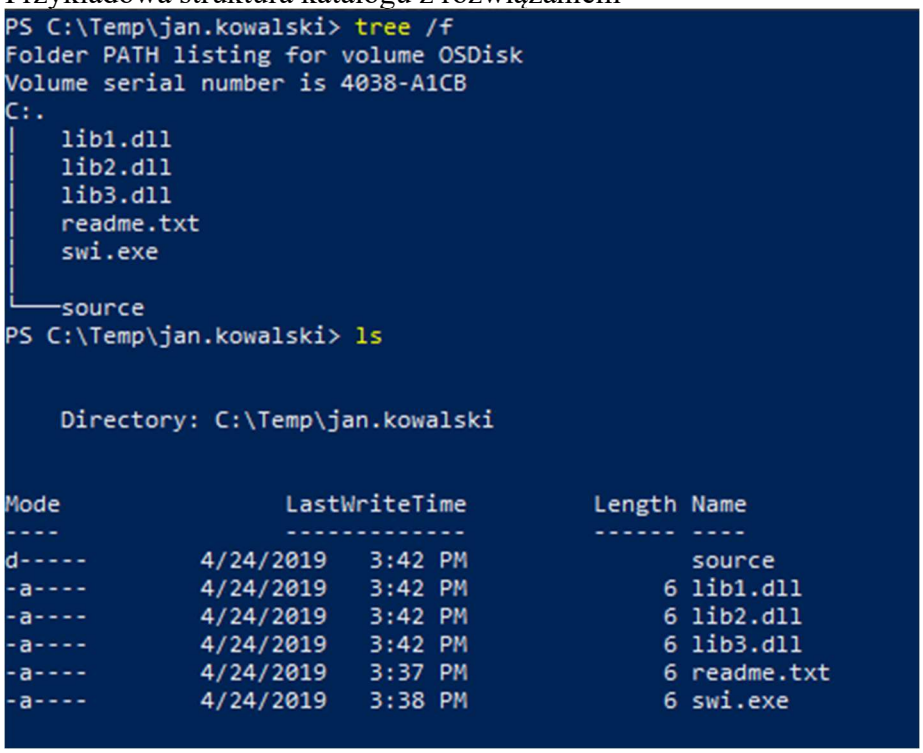
- ilość godzin spędzonych w biurze w danym dniu
- dodać literkę 'w' (weekend) jeżeli dzień był weekendem
- dodać literki 'ot' (overtime) jeżeli ktoś spędził w biurze więcej niż 9h
- dodać literki 'ut' ('undertime') jeżeli ktoś spędził w biurze mniej niż 6h
- 'i' (inconclusive) jeżeli uznasz, że obliczony czas pracy jest niejednoznaczny
- każdy ostatni dzień tygodnia (w biurze) ma dodatkowo zawierać sumę wszystkich godzin spędzonych w tygodniu w biurze, oraz podawać czas nadgodzin w formacie hh:mm:ss, bądź czas niedomiaru godzin spędzonych w tygodniu w biurze w formie -hh:mm:ss
- domyślny czas pracy dla jednego roboczego dnia to 8h

Przyłożenie identyfikatora do czytnika otwiera drzwi. Wszędzie, gdzie jest czytnik są też drzwi, zwykłe jak do mieszkania tyle że na zamek magnetyczny.
Biurowiec posiada wejście główne z czytnikiem identyfikatorów oraz wejście przez garaż. Dodatkowo na każdym piętrze znajduje się czytnik przy wejściu z części wspólnej na teren biura firmy na danym piętrze. Wyjście z biura jest możliwe poprzez otworzenie drzwi identyfikatorem, alternatywnie poprzez otworzenie drzwi przyciskiem. Wyjście z budynku możliwe jest tylko poprzez otwarcie drzwi identyfikatorem.

Zadanie należny dostarczyć w formie, która umożliwia jego uruchomienie bez wymaganych dodatkowych kroków (skompilowany plik binarny, bądź skrypt Python, wraz ze wszystkimi zależnościami jak np. dodatkowe biblioteki). Wszystkie pliki wymagane do uruchomienia oprogramowania powinny znajdować się w folderze o nazwie odpowiadającej formatowi imie.nazwisko autora oraz skompresowane "zipem", pliki źródłowe zadania (bądź cały projekt) powinny znajdować się w pod folderze 'source'. Jeżeli się zastanawiasz, jak nazwać projekt, luźną propozycją z naszej strony jest 'swi' (ale nie jest to wymagana nazwa). Jeżeli chcesz nam

coś przekazać, najlepiej umieścić wiadomość w pliku readme.txt (np. wersję Python której używałeś, czy cokolwiek innego). Po przygotowaniu pliku .zip wrzuć go np. na dysk google, one drive, drobpbox, github (jako release), czy jakiekolwiek inne medium do dzielenia się plikami. Pamiętaj, aby plik był udostępniony dla użytkownika praktyki@solarwinds.com, oraz wyślij do nas link do pobrania pliku. Nie wysyłaj pliku jako załącznik email, ponieważ wiadomość do nas nie dojdzie (a np. gmail nawet Ci na to nie pozwoli). Tekst w mailu z rozwiązaniem zadania będzie generalnie pomijany, więc zgodnie z wcześniejszą informacją, wszelkie informacje dotyczące oprogramowania umieć w pliku readme.txt.

Przykładowa struktura katalogu z rozwiązaniem



```
PS C:\Temp\jan.kowalski> tree /f
Folder PATH listing for volume OSDisk
Volume serial number is 4038-A1CB
C:.
|   lib1.dll
|   lib2.dll
|   lib3.dll
|   readme.txt
|   swi.exe
|
\---source
PS C:\Temp\jan.kowalski> ls


    Directory: C:\Temp\jan.kowalski


Mode                 LastWriteTime         Length Name
----                 -------------         ------ ----
d-----        4/24/2019   3:42 PM                source
-a----        4/24/2019   3:42 PM              6 lib1.dll
-a----        4/24/2019   3:42 PM              6 lib2.dll
-a----        4/24/2019   3:42 PM              6 lib3.dll
-a----        4/24/2019   3:37 PM              6 readme.txt
-a----        4/24/2019   3:38 PM              6 swi.exe
```

Przykładowy plik wejściowy

```
Date;Event;Gate
2019-02-04 09:05:58 ;Reader entry;E/0/KD1/7-9
2019-02-04 09:07:03 ;Reader entry;E/3/KD1/3-8
2019-02-04 17:32:34 ;Reader exit;E/3/KD1/3-8
2019-02-04 19:33:03 ;Reader exit;E/0/KD1/7-8
2019-02-05 09:10:34 ;Reader entry;E/0/KD1/7-9
2019-02-05 09:11:35 ;Reader entry;E/3/KD1/3-8
2019-02-05 15:26:36 ;Reader entry;E/0/KD1/8-8
2019-02-06 09:21:57 ;Reader entry;E/0/KD1/7-9
2019-02-06 09:22:27 ;Reader entry;E/3/KD1/3-8
2019-02-06 12:07:06 ;Reader exit;E/3/KD1/3-8
2019-02-06 12:07:54 ;Reader entry;E/0/KD1/7-8
2019-02-06 12:23:43 ;Reader entry;E/0/KD1/7-9
2019-02-06 12:24:31 ;Reader entry;E/3/KD1/3-8
2019-02-06 16:09:44 ;Reader exit;E/0/KD1/8-8
2019-02-07 09:09:57 ;Reader entry;E/0/KD1/7-9
```

```
2019-02-07 09:10:27 ;Reader entry;E/3/KD1/3-8
2019-02-07 10:56:26 ;Reader exit;E/3/KD1/3-8
2019-02-07 10:57:18 ;Reader entry;E/0/KD1/7-8
2019-02-07 11:05:01 ;Reader entry;E/0/KD1/7-9
2019-02-07 11:06:40 ;Reader exit;E/2/KD1/4-8
2019-02-07 12:33:01 ;Reader entry;E/3/KD1/3-8
2019-02-07 18:33:50 ;Reader exit;E/0/KD1/7-8
```

Na podstawie powyższych przykładowych danych powinien powstać plik wynikowy który zawiera

```
Day 2019-02-04 Work 10:27:05 ot
Day 2019-02-05 Work 6:16:02 i
Day 2019-02-06 Work 6:47:47
Day 2019-02-07 Work 9:23:53 ot 32:54:47 00:54:47
```

Osoba weryfikująca zadanie wykona następujące czynności

1. Rozpakuje plik imie.nazwisko.zip
2. Sprawdzi, czy jest pod katalog 'source'
3. Sprawdzi, czy jest plik readme.txt, jeżeli plik istnieje to z ciekawości go otworzy (w takim wypadku lepiej, żeby nie był pusty).
4. Wklei do katalogu weryfikacyjny plik "input.csv" (trochę dłuższy niż plik z zadania)
5. Uruchomi program*, sprawdzi plik wyjściowy (krok ma na celu weryfikacje czy oprogramowanie działa zgodnie z założeniami)
6. Wklei do katalogu swój, wcześniej przygotowany, plik weryfikacyjny "input.csv" (dodatkowy plik z przypadkami szczególnymi, ma na celu sprawdzenie czy oprogramowanie się 'wysypie')
7. Uruchomi program*, sprawdzi plik wyjściowy
   * program będzie uruchamiany z poziomu powłoki systemu (bash/powershell) (Windows Server, bądź Linux Debian 9)
8. Zapisze swoje spostrzeżenia

Jeżeli czegoś nie jesteś pewien/pewna, nie panikuj, sprawdź jeszcze raz instrukcje, jeżeli w instrukcji nie ma interesujących Cie informacji, **zrób to co podpowiada Ci zdrowy rozsądek.** Jedynym ograniczeniem jakie nakładamy na realizację zadania jest język programowania (możliwości wymienione na początku zadania), najważniejsze dla nas jest, aby oprogramowanie działało. Jeżeli pisanie zadania tak Cię będzie bawić, że będziesz myślał "co jeszcze by tu dopisać", śmiało możesz dopisać unit testy, testy End2End bądź stworzyć własne dodatkowe pliki wejściowe csv. Ale proszę nie próbuj pisać UI czy instalatora do oprogramowania.

Jeżeli weryfikacja oprogramowania zakończy się sukcesem, będziemy się kontaktować w sprawie umówienia rozmowy w formie zdalnej bądź w biurze Solarwinds w Krakowie (kompleks Bonarka 4 Business). Porozmawiamy wtedy z Tobą na temat Twojego programu oraz zostaniesz poproszona(y) o drobne rozszerzenie funkcjonalności (więc pamiętaj, żeby w pliku

imie.nazwisko.zip dostarczyć źródła, które się kompilują. W przypadku rozmowy zdalnej warto mieć źródła pod ręką).