Tematy projektów z Pythona

Języki Symboliczne 2016/17, studia stacjonarne

Spis treści

Ogólnie uwagi

1. Wymagana jest znajomość przedstawianego kodu.
2. W trakcie oddawania konieczna jest prezentacja przypadków użycia wymaganych przez prowadzącego, mieszczących się w ramach opisu zadania.
3. Wszystkie dane wprowadzane przez użytkownika powinny być sprawdzane, w razie wpisania niepoprawnych wartości powinna zostać wyświetlona wiadomość informująca użytkownika.
4. Do rysowania okienek polecana jest biblioteka tkinter (https://docs.python.org/3/library/tk.html).
5. Punktacja:
   a. 50% - poprawność funkcjonalna,
      - Seria testów zależna od tematu, podana razem z opisem tematu. Każdy zaliczony test daje $^{1}/_{N}$*50% punktów, gdzie N to liczba testów przewidzianych dla danego tematu.
   b. 40% - poprawne wykorzystanie czterech wybranych z poniższej listy konstrukcji Pythona (po 10%):
      - lambda-wyrażenia,
      - list comprehensions lub dictionary comprehensions,
      - dziedziczenie,
      - wyjątki (m.in. walidacja danych wprowadzanych przez użytkownika),
      - własne moduły (podział programu na przynajmniej dwa pliki: jeden do obsługi logiki i drugi do obsługi interfejsu).
   c. 10% - wyróżniające elementy, w szczególności:
      - Wykorzystanie zaawansowanych konstrukcji Pythona (np. generatory, metaklasy, dekoratory).
      - Wykorzystanie zaawansowanych algorytmów zapewniających dodatkową funkcjonalność ponad opisane minimum.
   d. Do każdego elementu oceny z punktów a, b i c może zostać zadane dodatkowe pytanie. Nieudzielenie odpowiedzi (lub udzielenie błędnej odpowiedzi) skutkuje brakiem punktów.
   e. Prowadzący zastrzegają sobie możliwość zmniejszenia liczby punktów z kategorii b i c (nawet do zera) w przypadku błędnego wykorzystania innych niż wymienione konstrukcji Pythona.
6. W razie jakichkolwiek wątpliwości (szczególnie dotyczących punktu 5e) należy skonsultować się z prowadzącym przed oddaniem projektu.
7. W ramach jednej grupy laboratoryjnej ten sam temat może być realizowany przez najwyżej dwie osoby. Wybór tematu musi zostać zaakceptowany przez prowadzącego grupę laboratoryjną.
8. Zadania należy oddawać na zajęciach w wyznaczonych przez prowadzącego terminach (około cztery osoby na jednym laboratorium). W przypadku usprawiedliwionej nieobecności przysługuje dodatkowy termin oddania projektu. Nieusprawiedliwiona nieobecność skutkuje oceną 2.0 za projekt.

9. W miarę dostępności prowadzących możliwe jest oddanie projektu przed wyznaczonym terminem. W takim przypadku możliwe będzie poprawienie uzyskanej oceny do wyznaczonego terminu oddania.
10. Możliwe jest zgłoszenie własnego tematu. Należy przesłać prowadzącemu propozycje opisu i testów.
11. Opisy projektów i testy mogą ulec niewielkim modyfikacjom mającym na celu usunięcie niejasności.

Tematy i opisy projektów

1. Automat biletowy MPK

Opis zadania

- Automat przechowuje informacje o monetach znajdujących się w nim (1, 2, 5, 10, 20, 50gr, 1, 2, 5zł) [dziedziczenie: można obsługiwać banknoty jako osobną klasę o wspólnej bazie z monetami]
- Okno z listą biletów w różnych cenach (jako przyciski). Wymagane bilety: 20-minutowy, 40-minutowy, 60-minutowy w wariantach normalnym i ulgowym.
- Możliwość wybrania więcej niż jednego rodzaju biletu. Możliwość wprowadzenia liczby biletów.
- Po wybraniu biletu pojawia się okno z listą monet (przyciski) oraz możliwością dodania kolejnego biletu lub liczby biletów.
- Interfejs ma dodatkowo zawierać pole na wybór liczby wrzucanych monet (domyślnie jedna).
- Po wrzuceniu monet, których wartość jest większa lub równa cenie wybranych biletów, automat sprawdza czy może wydać resztę.
  - Brak reszty/może wydać: wyskakuje okienko z informacją o zakupach, wydaje resztę (dolicza wrzucone monety, odlicza wydane jako reszta), wraca do wyboru biletów.
  - Nie może wydać: wyskakuje okienko z napisem "Tylko odliczona kwota" oraz zwraca włożone monety.

Testy

a. Bilet kupiony za odliczoną kwotę - oczekiwany brak reszty.
b. Bilet kupiony płacąc więcej - oczekiwana reszta.
c. Bilet kupiony płacąc więcej, automat nie ma jak wydać reszty - oczekiwana informacja o błędzie oraz zwrócenie takiej samej liczby monet o tych samych nominałach, co wrzucone.
d. Zakup biletu płacąc po 1gr - suma stu monet 1gr ma być równa 1zł (dla floatów suma sto razy 0.01+0.01+...+0.01 nie będzie równa 1.0). Płatności można dokonać za pomocą pętli for w interpreterze.
e. Zakup dwóch różnych biletów naraz - cena powinna być sumą.
f. Dodanie biletu, wrzucenie kilku monet, dodanie drugiego biletu, wrzucenie pozostałych monet, zakup za odliczoną kwotę - oczekiwany brak reszty (wrzucone monety nie zerują się po dodaniu biletu).
g. Próba wrzucenia ujemnej oraz niecałkowitej liczby monet (oczekiwany komunikat o błędzie).

2. Parkomat

Opis zadania

- Parkomat przechowuje informacje o monetach znajdujących się w nim (1, 2, 5, 10, 20, 50gr, 1, 2, 5zł) [dziedziczenie: można obsługiwać banknoty jako osobną klasę o wspólnej bazie z monetami]
- Okno z polem tekstowym na numer rejestracyjny pojazdu, aktualną datą (rok, miesiąc, dzień, godzina, minuta), datą wyjazdu z parkingu (rok, miesiąc, dzień, godzina, minuta - domyślnie aktualna data + 1 godzina), oraz przyciskiem "Zapłać"
- Zasady strefy parkowania:
  - Liczy się każda rozpoczęta godzina.
  - Pierwsza godzina płatna 2zł.
  - Każda kolejna godzina 4zł.
  - Parkowanie 24h kosztuje 100zł.
  - Parkowanie 48h kosztuje 250zł.
  - Parkowanie tydzień kosztuje 1800zł.
  - Parkowanie miesiąc (cztery tygodnie) kosztuje 7500zł.

  Dodatkowo:
  - Parkowanie od 8:00 do 8:01 - płatne za pierwszą godzinę.
  - Parkowanie od 8:00 do 9:01 - płatne za dwie godziny (pierwsza + kolejna).
  - Parkowanie 25h = cena parkowania 24h + cena pierwszej godziny.
  - Parkowanie 26h = cena parkowania 24h + cena kolejnej godziny + cena pierwszej godziny.
  - Parkowanie 72h = cena parkowania 48h + cena parkowania 24h.
  - Parkowanie 74h = cena parkowania 48h + cena parkowania 24h + cena kolejnej godziny + cena pierwszej godziny.
  - Parkowanie 8 dni = cena parkowania tydzień + cena parkowania 24h

  ... i tak dalej.
- Po wpisaniu rejestracji, wybraniu czasu wyjazdu (znajdującego się w przyszłości), obliczana jest cena do zapłaty (zgodnie z zasadami powyżej), i pojawia się okno wrzucania monet (przyciski) z opcją płacenia kartą.
  - Po wrzuceniu monet, których wartość jest większa lub równa obliczonej cenie, parkomat sprawdza czy może wydać resztę.
    - Brak reszty/może wydać: wyskakuje okienko z napisem "Drukowanie paragonu", wydaje resztę (dolicza wrzucone monety, odlicza wydane jako reszta), wraca do okna głównego.
    - Nie może wydać: wyskakuje okienko z napisem "Tylko odliczona kwota" oraz zwraca wrzucone monety.
  - Po wybraniu "Płatność kartą", wyskakuje okienko z napisem "Drukowanie paragonu". W losowych momentach (szansa 1%) zamiast tego wyskakuje informacja "Transakcja odrzucona".W obu przypadkach program wraca do okna głównego.

Testy

a. Parkowanie 1h za odliczoną kwotę - oczekiwany brak reszty.
b. Parkowanie 2h płacąc więcej - oczekiwana reszta,
c. Parkowanie 3h, automat nie ma jak wydać reszty - oczekiwana informacja o błędzie,
d. Próba wpisania daty w przeszłości - oczekiwana informacja o błędzie.
e. Próba wpisania niepoprawnej daty - oczekiwana informacja o błędzie.
f. Puste pole numeru rejestracyjnego - oczekiwana informacja o błędzie.
g. Parkowanie 1h płacąc po 1 gr - suma stu monet ma być równa 1zł (dla floatów suma sto razy 0.01+0.01+...+0.01 nie będzie równa 1.0). Płatności można dokonać za pomocą pętli for w interpreterze.
h. Parkowanie 27h - oczekiwana cena 110zł (24h + pierwsza + 2*kolejna).
i. Parkowanie 2 miesiące, 3 tygodnie, 5 dni, 7 godzin - oczekiwana cena 21’026zł.
j. Płatność kartą w przypadku z punktu i. - oczekiwana informacja o sukcesie.

Wygeneruj plik zawierający kolumny: liczba godzin, liczba miesięcy, liczba tygodni, liczba okresów 48h, liczba okresów 24h, liczba kolejnych godzin, liczba pierwszych godzin oraz cena parkowania.
Każdy wiersz powinien zawierać liczbę godzin przeliczoną na miesiące, tygodnie itd, oraz cenę do zapłacenia za parkowanie przez taki czas. Plik powinien zostać wygenerowany dla liczby godzin od 1 do 1344 (2 miesiące liczone po 4 tygodnie po 7 dni po 24 godziny) z krokiem co 1 godzinę.

3. Automat sprzedający napoje

Opis zadania

- Automat przechowuje informacje o monetach znajdujących się w nim (1, 2, 5, 10, 20, 50gr, 1, 2, 5zł) [dziedziczenie: można obsługiwać banknoty jako osobną klasę o wspólnej bazie z monetami]
- Automat przechowuje informacje o towarach znajdujących się w nim (przedmioty o numerach od 30 do 50), każdy o określonej cenie w określonej liczbie (domyślnie po 5 sztuk każdego towaru)
- Okno z przyciskami pozwalającymi na wrzucanie monet, polem wyświetlającym kwotę wrzuconych monet, przyciskiem przerywającym transakcję (wyrzuca wrzucone monety), przyciskami 0-9 pozwalającymi wpisać numer wybranego towaru oraz polem wyświetlającym wpisany numer towaru.
- Po wybraniu poprawnego numeru towaru:
  - Jeśli wrzucono za mało monet, wyskakuje okno z informacją o cenie towaru oraz (jeśli towar się skończył) o jego braku w automacie.
  - Jeśli wrzucono monety, których wartość jest większa lub równa cenie wybranego towaru, automat sprawdza czy towar jest dostępny i czy może wydać resztę
    - Brak towaru: wyskakuje okienko z informacją o braku w automacie.
    - Brak reszty/może wydać: wyskakuje okienko z informacją o zakupach, wydaje resztę (dolicza wrzucone monety, odlicza wydane jako reszta, odlicza wydany towar), odejmuje towar.
    - Nie może wydać: wyskakuje okienko z napisem "Tylko odliczona kwota".

Testy

a. Sprawdzenie ceny jednego towaru - oczekiwana informacja o cenie.
b. Wrzucenie odliczonej kwoty, zakup towaru - oczekiwany brak reszty.
c. Wrzucenie większej kwoty, zakup towaru - oczekiwana reszta.
d. Wykupienie całego asortymentu, próba zakupu po wyczerpaniu towaru - oczekiwana informacja o braku.
e. Sprawdzenie ceny towaru o nieprawidłowym numerze (<30 lub >50) - oczekiwana informacja o błędzie.
f. Wrzucenie kilku monet, przerwanie transakcji - oczekiwany zwrot monet.
g. Wrzucenie za małej kwoty, wybranie poprawnego numeru towaru, wrzucenie reszty monet do odliczonej kwoty, ponowne wybranie poprawnego numeru towaru - oczekiwany brak reszty.
h. Zakup towaru płacąc po 1 gr - suma stu monet ma być równa 1zł (dla floatów suma sto razy 0.01+0.01+...+0.01 nie będzie równa 1.0). Płatności można dokonać za pomocą pętli for w interpreterze.

4. Program do wyszukiwania połączeń kolejowych

Opis zadania

- Dwa okna:
  - Pierwsze: ustawianie połączeń (dwie kolumny przycisków opcji (radio button) lub listy rozwijane z nazwami miejscowości), przycisk "połącz", przycisk "rozłącz" oraz macierz istniejących połączeń (macierz sąsiedztwa grafu połączeń). Użytkownik wybiera miejscowości i tworzy między nimi połączenie.
  - Drugie: wyszukiwanie połączenia, dwie listy rozwijane, pierwsza oznaczająca miasto startowe, druga miasto docelowe, przycisk "szukaj", oraz lista przystanków na znalezionej trasie
- Program na początku ma zdefiniowane kilka (10) miast oraz połączeń między nimi (początkowo do każdego miasta da się dojechać, ale nie z każdego bezpośrednio do innego).
- Użytkownik wybiera miasto z którego chce dojechać, oraz miasto docelowe i wciska przycisk "szukaj":
  - Połączenie znalezione: wpisuje do listy kolejne przystanki, np: "Kraków -> Katowice", "Katowice -> Wrocław", "Wrocław -> Ankh-Morpork"
  - Połączenia brak: wyskakuje okienko z informacją "Brak połączenia na tej trasie"
- Program powinien realizować wyszukiwanie połączeń za pomocą algorytmu BFS. Graf połączeń powinien mieć dwie możliwe reprezentacje (do wyboru w pierwszym oknie): macierz sąsiedztwa albo listy sąsiedztwa, realizowane jako dwie klasy dziedziczące po klasie AbstractGraph definiującej niezbędny interfejs. Zmiana reprezentacji powinna być możliwa w dowolnym momencie działania programu.

Testy

a. Wyszukanie połączenia bezpośredniego między dwoma miastami.
b. Wyszukanie połączenia między dwoma miastami z przynajmniej dwoma przystankami pośrednimi.
c. Dodanie połączenia bezpośredniego między dwoma miastami z punktu b. - oczekiwana informacja o połączeniu bezpośrednim.
d. Usunięcie połączeń bezpośrednich między miastami z punktu a., wyszukanie połączenia - oczekiwana informacja o przesiadce.
e. Usunięcie wszystkich połączeń do danego miasta - oczekiwana informacja o braku połączenia.
f. Wykonanie testu b przy drugiej reprezentacji grafu.
g. Wykonanie testu c przy drugiej reprezentacji grafu.

5. Warcaby

Opis zadania

- Okno z siatką przycisków 8x8 oraz przyciskiem do resetowania gry.
- Przyciski reprezentują pola planszy do gry w warcaby. Pola puste - przyciski bez tekstu. Pola z pionkami gracza 1 - przycisk z tekstem “C”. Pola z pionkami gracza 2 - przycisk z tekstem “B”. Damki oznaczane są dodatkową literą d (“Cd”, “Bd”).
- Nad planszą wyświetlana jest informacja “Tura gracza 1” lub “Tura gracza 2”.
- Gracz wybiera pionka (tekst pola zmienia się z “C” na “[C]” lub z “B” na “[B]”), a potem pole na które chce wykonać ruch. Jeśli ruch jest dozwolony, pionek jest przestawiany. Jeśli nie, to wyświetlany jest komunikat “ruch niedozwolony”.
- Zasady jak w warcabach (https://pl.wikipedia.org/wiki/Warcaby, dowolny wariant). Zwykłe pionki i damki mają być obiektami dwóch różnych klas dziedziczących po klasie Pionek.
- Gdy gra się kończy, wyświetlane jest okienko z napisem “Wygrał gracz 1” lub “Wygrał gracz 2”, zależnie kto wygrał grę. Możliwe jest zresetowanie planszy bez zamykania głównego okna.

Testy

a. Wykonanie po dwa ruchy przez każdego z graczy.
b. Niepowodzenie błędnego ruchu pionkiem.
c. Wykonanie bicia pojedynczego pionka.
d. Wykonanie bicia przynajmniej dwóch pionków.
e. Zamiana pionka w damkę.
f. Bicie damką.
g. Wygrana gracza grającego czarnymi pionkami.
h. Rozpoczęcie nowej gry po zwycięstwie jednego z graczy.

Wskazane jest przygotowanie specjalnych początkowych rozstawień pionków dla testów d, e, f, g, h.

6. Cztery w rzędzie (https://en.wikipedia.org/wiki/Connect_Four)

Opis zadania

- Okno wyświetlające siatkę 7 kolumn x 6 wierszy, przycisk nad każdą kolumną, informację “Tura gracza 1” lub “Tura gracza 2”, przycisk do resetowania gry oraz rozwijalną listę wyboru reguł gry.
- Początkowo pola siatki są puste.
- Gracze na zmianę wrzucają monety do wybranych przez siebie kolumn.
- Pola w których jest moneta gracza 1 są czerwone, pola z monetami gracza 2 są żółte (tkinter, Canvas, http://stackoverflow.com/a/12254442).
- Gracze wybierają kolumnę klikając przycisk nad nią.
- Wygrywa gracz który pierwszy ustawi cztery monety w linii (poziomo, pionowo lub po skosie).
- Gdy gra się kończy, wyświetlane jest okienko z napisem “Wygrał gracz 1” lub “Wygrał gracz 2”, zależnie kto wygrał grę. Możliwe jest zresetowanie planszy bez zamykania głównego okna.
- Reprezentacja reguł gry ma być realizowana poprzez hierarchię klas. Klasa bazowa definiuje między innymi funkcję wirtualną ktoWygral() nadpisywaną w klasach pochodnych. Realizowane powinny być przynajmniej dwa zestawy reguł, jako dwie klasy pochodne.

Testy

a. Wykonanie po dwa ruchy przez każdego z graczy - monety spadają na dół pola gry lub zatrzymują się na już wrzuconym żetonie.
b. Ułożenie pionowej linii monet przez jednego gracza - oczekiwana informacja o jego wygranej.
c. Ułożenie poziomej linii monet przez drugiego gracza - oczekiwana informacja o jego wygranej.
d. Ułożenie skośnej linii przez dowolnego gracza - oczekiwana informacja o jego wygranej.
e. Zapełnienie pola gry tak, że żaden gracz nie ułożył linii - oczekiwana informacja o remisie.
f. Ułożenie linii dłuższej niż 4 przez jednego z graczy - oczekiwana informacja o jego wygranej.
[c][c][c][ ][c][c][c]
[ż][ż][ż][ ][ż][ż][ż] <- w następnym ruchu gracz żółty wrzuci monetę w środkową kolumnę.
g. Próba wrzucenia monety do zapełnionej kolumny - oczekiwana informacja o błędzie.

7. Mastermind (https://en.wikipedia.org/wiki/Mastermind_(board_game))

Opis zadania

- Okno z polem tekstowym na 4 cyfry, listą odpowiedzi, przyciskiem “Sprawdź”, przyciskiem “Oszust!” oraz przyciskiem “Reset”.
- Po rozpoczęciu gry generowana jest losowa liczba (kod) złożona z czterech cyfr od 1 do 6 włącznie (1111, 1112, 1113, …, 3455, 3456, 3461, 3462, …, 6665, 6666).
- Gracz wpisuje cztery cyfry od 1 do 6 do pola tekstowego i naciska przycisk “Sprawdź”.
- Do pola odpowiedzi dopisywana jest odpowiedź zawierająca: liczbę wpisaną przez gracza, liczbę cyfr na poprawnych pozycjach oraz liczbę cyfr występujących w kodzie, ale na złych pozycjach.
- Jeśli gracz wpisał liczbę będącą kodem, wyświetlane jest okno z napisem “Wygrana”.
- Jeśli gracz po 12 próbach nie odgadł kodu, wyświetlane jest okno z napisem “Przegrana”.
- Logika gry powinna być realizowana przez osobną klasę, dziedziczącą po klasie RegulyGry. Z klasy RegulyGry powinna być również wydziedziczona druga klasa, generująca niepoprawne odpowiedzi. Wybór zestawu reguł powinien być dokonywany losowo przed każdą grą.
- Jeśli gracz nacisnął przycisk “Oszust!” przy poprawnych regułach gry, program powinien wyświetlić okno z napisem “Tere fere.” oraz wylosowanym kodem.
- Jeśli gracz nacisnął przycisk “Oszust!” przy niepoprawnych regułach gry, program powinien wyświetlić okno z napisem “Złapałeś/łaś mnie!”

Testy

a. Wyświetlenie (wypisanie w konsoli) wylosowanego kodu, wpisanie odpowiedzi z błędnymi cyframi - oczekiwana informacja o braku poprawnych trafień.
b. Wyświetlenie wylosowanego kodu, wpisanie odpowiedzi z poprawnymi cyframi w złych miejscach - oczekiwana informacja o niepoprawnym położeniu.
c. Wyświetlenie wylosowanego kodu, wpisanie odpowiedzi z dwoma poprawnymi cyframi w dobrych miejscach i dwoma poprawnymi w złych miejscach - oczekiwana informacja o dwóch trafieniach i dwóch złych pozycjach.
d. Wyświetlenie wylosowanego kodu, wpisanie poprawnej odpowiedzi - oczekiwana informacja o wygranej.
e. Wpisanie 12 razy niepoprawnego kodu - oczekiwana informacja o przegranej
f. Próba wpisania niepoprawnego kodu do pola odpowiedzi (mniej lub więcej niż 4 znaki, znaki nie będące cyframi od 1 do 6) - oczekiwane nieuznanie kodu (gracz nie traci tury).
g. Wciśnięcie przycisku “Oszust” przy poprawnych zasadach gry - oczekiwana informacja “tere fere”.

h. Wciśnięcie przycisku “Oszust” przy niepoprawnych zasadach gry - oczekiwana informacja o oszukiwaniu przez komputer.
i. Wpisanie 10 kodów, resetowanie gry, wpisanie 5 kodów - oczekiwane normalne działanie gry (czy licznik tur resetuje się po wciśnięciu “Reset”).

8. Symulacją obsługi klientów przy okienkach z wyliczaniem statystyk

Opis zadania

- Trzy okienka (na przykład w urzędzie) obsługują klientów przychodzących w jednej z dwóch spraw (A, B).
- Okienka reprezentowane są przez obszary w dole okna programu z wyborem (checkbox) spraw jakie są tam załatwiane i przyciskiem “Następny”.
- Nowe osoby chcące coś załatwić reprezentowane są przez wciśnięcia jednego z czterech przycisków na górze okna programu (“A”, “B”, “VIP A”, “VIP B”). Obok przycisków napisana jest liczba czekających klientów z daną sprawą.
- Program automatycznie przydziela osoby do okienek w momencie naciśnięcia przycisku “Następny”. Osoby są przyjmowane w kolejności przyjścia, najpierw z kolejek VIP, a następnie (gdy kolejka VIP jest pusta) ze zwykłych kolejek. Jeśli zwykłe kolejki też są puste, to następny pasujący klient który przyjdzie przyjmowany jest bez czekania.
- Jeśli klient VIP czeka dłużej niż 10 sekund, powinno się wyświetlić okno z tekstem “To skandaliczne!”.
- Klienci są reprezentowani przez klasy KlientZwykly i KlientVIP dziedziczące po klasie Klient. Posiadają one metody wejscie, wyjscie i tick odpowiednio wywoływane przy wejściu klienta, jego wyjściu i co sekundę, rejestrujące odpowiednie czasy i reagujące na zdarzenia.
- Program posiada również w głównym oknie przycisk “Zakończ” wyświetlający statystyki (średni czas oczekiwania w poszczególnych okienkach i w ogóle z rozbiciem na sprawy i bez) jak poniżej i resetujący stan. Wyświetlone mają być dwie tabele, osobno dla zwykłych klientów i klientów VIP.

---	Okienko 1	Okienko 2	Okienko 3	Łącznie
Sprawa A				
Sprawa B				
Łącznie				

Testy

a. Ustawienie okienka pierwszego na obsługę spraw A, drugiego na obsługę spraw B, a trzeciego na obsługę obu typów spraw (dotyczy wszystkich podpunktów o ile nie wyszczególniono inaczej). Wpuszczenie klienta ze sprawą A, naciśnięcie obsługi przy okienku drugim, naciśnięcie obsługi przy okienku pierwszym, wpuszczenie klienta ze sprawą B (powinien od razu być załatwiony w drugim okienku), zakończenie.
b. Wpuszczenie klienta A, wpuszczenie klienta VIP A, następny w okienku pierwszym, następny w okienku trzecim. Czasy obsługi powinny wskazywać na to, że klient VIP był obsłużony jako pierwszy.

c. Wpuszczenie klienta VIP A, wpuszczenie klienta B, następny w okienku trzecim, następny w okienku drugim, zakończenie.
d. Wpuszczenie klienta VIP B, wpuszczenie klienta A, następny w okienku trzecim, następny w okienku pierwszym, zakończenie.
e. Wpuszczenie klienta A, zakończenie (podsumowanie powinno wskazywać czas od wejścia klienta do zakończenia).
f. Wpuszczenie klienta A, wpuszczenie klienta VIP B, następny w okienku pierwszym, odczekanie do momentu wyświetlenia komunikatu o zbyt wolnej obsłudze.
g. Wpuszczenie po czterech klientów każdego rodzaju i obsługa ich wszystkich w okienkach pierwszym i drugim. Zakończenie.

9. Symulator kasjera

Opis zadania

- Okno podzielone jest na dwie części:
  - Lewa: pusta, pojawiają się tam towary do skasowania
  - Prawa: przyciski od 0 do 9, przycisk “backspace”, przycisk “Wyczyść”, przycisk “Zważ” oraz pole tekstowe - wciskanie przycisków cyfr powoduje dopisywanie ich do pola, backspace wymazuje ostatnio wpisaną cyfrę, przycisk “Wyczyść” czyści pole tekstowe.
    Jeśli zawartość pola tekstowego to 1 i wciśnięty został przycisk cyfry, pole tekstowe jest czyszczone i dopisywana jest tam wybrana cyfra.
- Na początku po lewej stronie znajduje się przycisk “Następny klient”, wciśnięcie go rozpoczyna grę (zapisanie aktualnego czasu i wyzerowanie licznika towarów)
- W losowym miejscu lewej strony okna pojawia się przycisk z:
  - nazwą towaru i jego liczebnością, np. “Arbuz x10”. Wciśnięcie przycisku powoduje zmniejszenie liczebności towaru o tyle, ile wpisane jest w polu tekstowym po prawej stronie, a pole to jest czyszczone (zawartość ustawiana na 1) lub
  - nazwą towaru i napisem “?kg” oznaczającym, że jest to towar do zważenia. Jego kasowanie polega na kliknięciu przycisku z towarem, następnie kliknięciu przycisku “Zważ”, a następnie ponownym kliknięciu przycisku z towarem. Po naciśnięciu przycisku “Zważ” etykieta towaru powinna się zmienić (ma być podana losowa waga od 0.05 do 2 kg).
- Jeśli wartość pola tekstowego po prawej stronie przewyższa liczebność towaru, to gracz przegrywa i wyświetlane jest okno informujące o tym.
- Po skasowaniu towaru (liczebność spadła do 0), do licznika towarów dodawana jest oryginalna liczebność towaru, oraz generowany jest kolejny towar.
- Generowane jest od 10 do 20 towarów, połowa z nich na sztuki. Liczba sztuk ma wynosić od 1 do 50 (losowo). Prawdopodobieństwo wylosowania liczebności 1 (pojedynczy artykuł) ma wynosić 50%.
- Po skasowaniu wszystkich towarów wyświetlany jest średni czas kasowania jednego przedmiotu (towar w liczebności 10 liczy się za 10 przedmiotów).
- Nazwy towarów mają być losowane z minimum dwudziestoelementowej listy.
- Towary mają być reprezentowane przez obiekty TowarNaSztuki i TowarNaWagę dziedziczących po klasie Towar. W obiektach ma być przechowywana nazwa towaru, liczba sztuk lub waga, czas pojawienia się w lewej części okna i czas skasowania.

Testy

a. Skasowanie towaru na sztuki po klikając na niego kilka razy.
b. Skasowanie towaru na sztuki wpisując jego liczność i klikając raz. Wymagane jest resetowanie pola do wartości 1.

c. Próba skasowania towaru na sztuki wpisując zbyt dużą liczność (oczekiwana informacja o przegranej).
d. Próba zważenia towaru na sztuki (oczekiwana informacja o przegranej).
e. Próba skasowania towaru na wagę jakby był towarem na sztuki (oczekiwana informacja o przegranej).
f. Skasowanie wszystkich towarów (oczekiwane okno z podsumowaniem symulacji).
g. Pokazanie, że liczebność 1 występuje odpowiednio często.

10. Program w Pythonie wywołujący funkcję z biblioteki dll wykonującą operacje na rysunku

Opis zadania

- Program operuje na rysunkach w odcieniach szarości (0-255) przy użyciu podanej maski (różne filtry).
- Okno z polem wyświetlającym rysunek, przyciskiem pozwalającym wybrać rysunek (wczytywanie z pliku), siatką 3x3 pól tekstowych (maska) umożliwiających wprowadzenie maski, oraz przyciskiem “Filtruj”.
- Wykorzystywane są toroidalne warunki brzegowe (np. prawymi sąsiadami pikseli na prawym brzegu są piksele z lewego brzegu).
- Użytkownik wczytuje rysunek, ustawa maskę i wciska przycisk “Filtruj”.
- Rysunek przesyłany jest do funkcji z pliku .dll, która stosuje podaną maskę na rysunku.
- Po zakończeniu działania funkcji, rysunek w programie powinien przedstawiać efekt działania filtru.
- Ponowne wciśnięcie przycisku “Filtruj” powinno działać na zmodyfikowanym rysunku.
- Do wywoływania funkcji z dll należy stosować bibliotekę ctypes (https://docs.python.org/3.6/library/ctypes.html).
- Do obsługi obrazów można stosować bibliotekę pillow (https://python-pillow.org/).

Testy

a. Wczytanie i wyświetlenie rysunku.
b. Filtrowanie z maską [[0, 0, 0], [0, 1, 0], [0, 0, 0]] z wykorzystaniem dll.
c. Filtrowanie z maską [[1, 0, -1], [2, 0, -2], [1, 0, -1]] (filtr Sobela), z wykorzystaniem dll.
d. Filtrowanie z maską [[1/9, 1/9, 1/9], [1/9, 1/9, 1/9], [1/9, 1/9, 1/9]] (rozmycie), z wykorzystaniem dll.
e. Wczytanie innego rysunku bez zamykania programu.
f. Próba wczytania pliku dll (oczekiwany komunikat o błędzie, możliwy ma być wybór innego pliku bez ponownego uruchamiania programu).
g. Wykonanie filtrów b, c i d kolejno na tym samym rysunku (rysunek -> filtr b -> rysunek z filtrem b -> filtr c -> rysunek z filtrami b i c -> filtr d -> rysunek z filtrami b, c, d)

11. Program w Pythonie wywołujący funkcję z biblioteki dll liczącą iloczyn macierzy

Opis zadania

- Okno z przyciskiem “Uruchom obliczenia”, polem z informacjami o ostatnim uruchomieniu (czas obliczeń w Pythonie, czas obliczeń w C/C++, czas potrzebny na konwersję danych), oraz polami pozwalającymi wybrać:
  - Liczbę kolumn macierzy,
  - Liczbę wierszy macierzy,
  - Zakres generowanych wartości w macierzach (od, do),
  - Rodzaj generowanych wartości (liczby całkowite/rzeczywiste),
  - Liczba powtórzeń mnożenia (do uśrednienia czasu obliczeń).
- Powinna dodatkowo istnieć możliwość ręcznego wprowadzenia niewielkich (do trzech wierszy i kolumn) macierzy które mają być przemnożone. Wprowadzanie tych macierzy można zrealizować przez tablice pól tekstowych do wprowadzania elementów macierzy.
- Po wciśnięciu przycisku “Uruchom obliczenia” generowane są dwie macierze o podanych przez użytkownika parametrach, które są następnie mnożone w Pythonie oraz w C/C++ (kod mnożący wywoływany z pliku .dll). Jeśli macierzy nie da się przemnożyć, ma zostać wyświetlony odpowiedni komunikat w oknie dialogowym (proszę użyć mechanizmu wyjątków).
- Implementacje w Pythonie i w C lub C++ mają być realizowane w dwóch klasach (MnozenieMacierzyCpp, MnozenieMacierzyPython) dziedziczących po klasie MnozenieMacierzy z metodą mnoz.
- Zapisywany jest czas obliczeń w pierwszym i drugim języku oraz czas przygotowania danych do przesłania do C/C++.
- Wyniki wpisywane są do pola z informacjami.
- Do wywoływania funkcji z dll należy stosować bibliotekę ctypes (https://docs.python.org/3.6/library/ctypes.html).

Testy

a. Wykonanie mnożenia (w Pythonie i przez dll):

$$\begin{bmatrix}1 & 0\\0 & 1\end{bmatrix} \times \begin{bmatrix}2 & 3\\4 & 5\end{bmatrix} = \begin{bmatrix}2 & 3\\4 & 5\end{bmatrix}$$

b. Wykonanie mnożenia (w Pythonie i przez dll):

$$\begin{bmatrix}1 & 2 & 1\\4 & -2 & 0\end{bmatrix} \times \begin{bmatrix}0 & 1\\2 & 0\\-1 & -1\end{bmatrix} = \begin{bmatrix}-1 & 0\\-4 & 4\end{bmatrix}$$

c. Wykonanie mnożenia (w Pythonie i przez dll):

$$\begin{bmatrix}1 & 2\\4 & -2\\0 & 3\end{bmatrix} \times \begin{bmatrix}0 & 1 & -3\\2 & -1 & 0\end{bmatrix} = \begin{bmatrix}4 & -1 & -3\\-4 & 6 & -12\\6 & -3 & 0\end{bmatrix}$$

d. Próba wykonania mnożenia macierzy 2x3 przez macierz 1x2 (oczekiwany komunikat o błędzie i możliwość kontynuowania pracy z programem bez ponownego uruchamiania.
e. Uzyskanie porównania czasów dla mnożenia macierzy 1x1, 10x10 i 50x50 (odpowiednio 100000, 1000 i 50 powtórzeń, liczby całkowite).
f. Uzyskanie porównania czasów dla mnożenia macierzy 200x200 (3 powtórzenia, liczby rzeczywiste).