RAPORT PROJEKT SO

**Fryc Bartłomiej, nr indeksu 151861**

**Założenia projektu nr 15 – Przychodnia**

**1. Organizacja pracy lekarzy**

• W przychodni pracuje dwóch lekarzy POZ oraz specjaliści: kardiolog, okulista, pediatra, lekarz medycyny pracy.

• Każdy lekarz w danym dniu ma określony limit przyjęć:

• Lekarze POZ: X1 pacjentów.

• Kardiolog: X2 pacjentów.

• Okulista: X3 pacjentów.

• Pediatra: X4 pacjentów.

• Lekarz medycyny pracy: X5 pacjentów.

• Lekarze POZ przyjmują około 60% wszystkich pacjentów (wartość losowa).

• Specjaliści przyjmują około 10% pacjentów każdy (wartość losowa).

• Do lekarzy POZ obowiązuje jedna wspólna kolejka, a do każdego specjalisty – indywidualna.

**2. Kierowanie pacjentów na dodatkowe badania**

• Lekarze POZ kierują około 20% swoich pacjentów (z X1) do specjalistów.

• Rejestracja do specjalisty odbywa się przez lekarza POZ.

• Jeśli specjalista nie ma wolnych miejsc, dane pacjenta (ID, skierowanie, lekarz wystawiający) zapisuje się w raporcie dziennym.

• Specjaliści mogą skierować około 10% swoich pacjentów na badania ambulatoryjne.

• Po badaniu pacjent wraca do specjalisty i wchodzi do gabinetu poza kolejnością.

**3. Zasady działania przychodni**

• Godziny pracy: od Tp do Tk.

• W przychodni może przebywać jednocześnie maksymalnie N pacjentów.

• Pozostali pacjenci czekają na zewnątrz.

• Dzieci poniżej 18 roku życia muszą być pod opieką osoby dorosłej.

• Każdy pacjent musi zarejestrować się przed wizytą u lekarza.

• Przychodnia posiada 2 okienka rejestracji:

• Zawsze działa co najmniej 1 okienko.

• Drugie okienko otwiera się, gdy w kolejce do rejestracji stoi więcej niż K pacjentów (K ≥ N/2).

• Drugie okienko zamyka się, gdy liczba pacjentów w kolejce spada poniżej N/3.

• Pacjenci VIP (np. honorowi dawcy krwi) mają prawo wejścia do lekarza poza kolejnością.

**4. Ograniczenia i limity**

• Pacjenci nie są przyjmowani, jeśli limit przyjęć do danego lekarza został wyczerpany.

• Jeśli w danym dniu wszystkie limity przyjęć zostały wyczerpane, pacjenci nie są wpuszczani do budynku.

• Pacjenci oczekujący w kolejce do lekarza na koniec dnia zostają przyjęci, ale nie mogą być kierowani na dodatkowe badania.

• Pacjenci oczekujący w kolejce do rejestracji na koniec dnia nie są przyjmowani, a ich dane zapisuje się w raporcie dziennym (ID, skierowanie, lekarz wystawiający).

**5. Działania na polecenie Dyrektora**

• **Sygnał 1**: Wybrany lekarz kończy pracę po zbadaniu bieżącego pacjenta, dane nieprzyjętych pacjentów zapisuje się w raporcie dziennym.

• **Sygnał 2**: Wszyscy pacjenci natychmiast opuszczają budynek.

**6. Raport dzienny**

• Zawiera dane pacjentów, którzy nie zostali przyjęci:

• ID pacjenta.

• Skierowanie (do kogo, kto wystawił).

• Powód nieprzyjęcia (np. brak miejsc, zamknięcie przychodni).

**Analiza procesu biznesowego jakim jest przychodnia (notacja BPMN)**

**Obraz zawierający diagram, szkic, Plan, Rysunek techniczny

Opis wygenerowany automatycznie**

**Mechanizmy Komunikacji**

**KOLEJKI KOMUNIKATOW**

**Obraz zawierający tekst, zrzut ekranu, Czcionka

Opis wygenerowany automatycznie**

Kolejki są używane do przekazywania procesu pacjenta pomiędzy różnymi instancjami

Pojawienie się -> kolejka zewnętrzna -> kolejka rejestracja -> kolejka do konkretnego lekarza w zależności od bycia VIPem bądź nie (opcjonalnie -> przekazanie do kolejki badań i powrót kolejką do konkretnego lekarza)

**PAMIĘĆ WSPÓŁDZIELONA**

Obraz zawierający tekst, Czcionka, zrzut ekranu

Opis wygenerowany automatycznie



Używana do przetrzymywania wartości dwóch zmiennych miedzy procesowych:

1. Liczba osób – liczba osób, która jest w przychodni w danym momencie
2. Suma kolejek – liczba osób, które są po rejestracji do konkretnych lekarzy, pozwalająca na zamknięcie przychodni w momencie przekroczenia sumy przyjęć wszystkich lekarzy.
3. Przetrzymywanie liczby osób w kolejce do lekarzy.

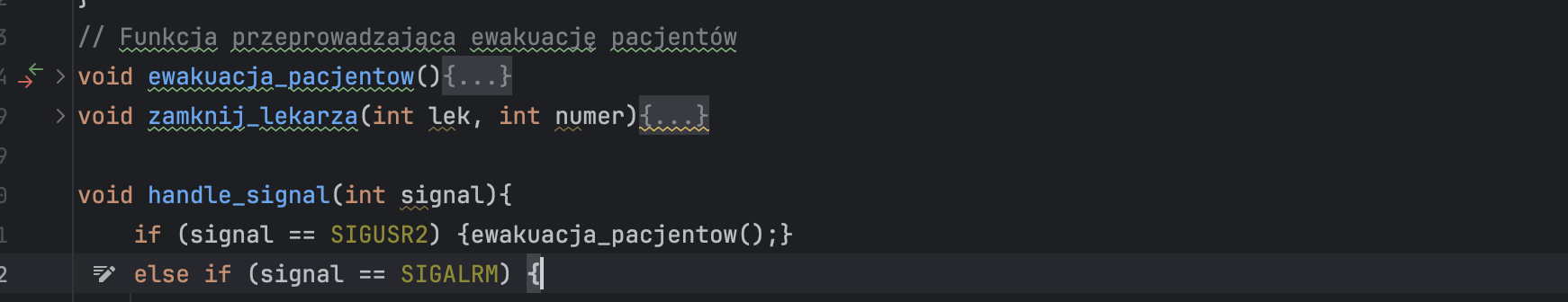
**SEMAFORY**

Obraz zawierający tekst, Czcionka, zrzut ekranu

Opis wygenerowany automatycznie

1. Semafor do rejestracji – do dwóch okienek przy pobieraniu z kolejki rejestracja.
2. Semafor liczba\_osob – żeby nie było konfliktu przy dodawaniu/odejmowaniu osob przy wchodzeniu i wychodzeniu z przychodni.
3. Semafor suma\_kolejka – jak wyżej, rejestracja dwóch osób równocześnie.
4. Semafor dostępu do poszczególnych kolejek lekarzy

**SYGNAŁY**

****

1. Sygnal ewakuacji wszystkich pacjentow – następnie mogą wchodzić ponownie
2. Zamknij lekarza – zamyka losowego lekarza

**PROCEDURY**

**Procedura main.c**

1. **Definicje zmiennych globalnych**
   1. Definiuje identyfikatory pamięci współdzielonej (shm\_id, shm\_id2).
   2. Tworzy wskaźniki do obsługi pamięci współdzielonej (liczba\_osob, suma\_kolejek).
   3. Deklaruje semafory (semafor\_rejestracja, semafor\_liczba\_osob, semafor\_suma\_kolejek).
   4. Zmienna zasoby\_wyczyszczone śledzi stan czyszczenia zasobów.
2. **Losowanie danych pacjenta**
   1. Funkcja losuj\_pacjenta() generuje dane pacjenta, w tym:
   2. Identyfikator pacjenta.
   3. Wiek (losowy od 1 do 100).
   4. Priorytet (np. VIP dla osób starszych z 30% szansą).
   5. Informację o obecności rodzica (dla osób poniżej 18 lat).
   6. Preferowanego lekarza (60% szans na POZ, pozostałe 40% równomiernie na specjalistów).
   7. PID procesu.
3. **Obsługa sygnałów i czyszczenie zasobów**
   1. Funkcja cleanup\_on\_exit():
   2. Zamyka procesy potomne.
   3. Usuwa semafory i pamięć współdzieloną.
   4. Zabezpiecza przed wielokrotnym czyszczeniem tych samych zasobów.
   5. Funkcja signal\_handler() obsługuje sygnały systemowe (SIGINT, SIGTERM) i wywołuje cleanup\_on\_exit().
4. **Tworzenie pamięci współdzielonej**
   1. pamiec\_wspoldzielona() oraz pamiec\_wspoldzielona2():
   2. Tworzą segmenty pamięci współdzielonej do przechowywania liczby osób w przychodni i sumy kolejek.
   3. Inicjalizują pamięć wartościami początkowymi (0).
5. **Inicjalizacja semaforów**
   1. Tworzone są semafory do kontroli dostępu do pamięci współdzielonej oraz rejestracji:
   2. semafor\_liczba\_osob – do zarządzania liczbą osób w przychodni.
   3. semafor\_rejestracja – do obsługi rejestracji.
   4. semafor\_suma\_kolejek – do zarządzania kolejkami.
6. **Tworzenie procesów**
   1. Procesy główne i potomne:
   2. **Rejestracja**: Tworzony proces obsługujący rejestrację pacjentów.
   3. **Lekarze POZ i specjaliści**: Uruchamiane są osobne procesy dla dwóch lekarzy POZ i czterech specjalistów.
   4. **Pacjenci**: Tworzone są procesy pacjentów, symulujących wizyty w przychodni.
   5. Procesy lekarzy i pacjentów są uruchamiane za pomocą fork() i execl().
7. **Obsługa zakończonych procesów**
   1. Funkcja process\_cleaner() oczekuje na zakończenie procesów potomnych, usuwając ich informacje z systemu.
8. **Główna pętla programu**
   1. Symuluje działanie przychodni poprzez uruchamianie procesów w ustalonym porządku.
   2. Procesy są tworzone w odstępach czasowych (sleep(1)), by symulować przybywanie pacjentów.
9. **Kończenie programu**
   1. Po zakończeniu wszystkich procesów i wątków następuje czyszczenie zasobów za pomocą cleanup\_on\_exit().

**Procedura rejestracja.c**

1. **Funkcje główne**
   1. rejestracja(int id):
      1. Odpowiada za przyjmowanie pacjentów z kolejki rejestracyjnej i kierowanie ich do odpowiednich kolejek lekarzy.
      2. Wykorzystuje kolejki komunikatów (np. KOLEJKA\_POZ, KOLEJKA\_KARDIOLOG, itd.) do przesyłania pacjentów.
      3. Sprawdza dostępność miejsc dla danego lekarza, w razie braku miejsca zapisuje pacjenta do raportu dziennego.
      4. Obsługuje priorytety pacjentów (VIP) i kieruje ich do odpowiednich kolejek VIP.
   2. zarzadz\_kolejka\_zewnetrzna():
      1. Obsługuje pacjentów oczekujących w kolejce zewnętrznej.
      2. Sprawdza, czy pacjenci mogą wejść do przychodni w zależności od liczby dostępnych miejsc.
      3. W razie braku miejsca pacjenci pozostają w kolejce zewnętrznej.
   3. zarzadz\_i\_monitoruj\_rejestracje():
      1. Uruchamia główny proces rejestracji oraz monitoruje liczbę pacjentów w kolejce rejestracyjnej.
      2. Jeśli liczba pacjentów przekracza próg (MAX\_OSOB\_W\_PRZYCHODNI / 2), uruchamia drugie okienko rejestracji.
      3. Gdy liczba pacjentów spada poniżej MAX\_OSOB\_W\_PRZYCHODNI / 3, zamyka drugie okienko.
2. **Obsługa sygnałów**
   1. signal(SIGUSR1, rejestracja\_end): Odbiera sygnał zakończenia pracy i ustawia flagę end\_rejestracja.
   2. Funkcja kill wysyła sygnały do procesów w celu zamknięcia drugiego okienka rejestracji.
3. **Zapis do raportu dziennego**
   1. zapisz\_do\_raportu(RaportPacjenta pacjent):
   2. Zapisuje dane pacjenta, który nie został przyjęty, do pliku tekstowego rapodrt\_zienny.txt.
4. **Główna pętla programu**
   1. Obsługuje pacjentów w kolejkach zewnętrznych i rejestracyjnych.
   2. Wysyła pacjentów do odpowiednich kolejek lekarzy na podstawie ich specjalizacji i priorytetów.
   3. Monitoruje stan kolejek i dostępność miejsc w przychodni.
5. **Multithreading i procesy**
   1. Wykorzystuje wątki (pthread\_create) do zarządzania kolejką zewnętrzną.
   2. Używa fork() do tworzenia procesów rejestracji (główne i drugie okienko).

**Kluczowe funkcjonalności:**

• Zarządzanie kolejkami pacjentów i ich priorytetami.

• Synchronizacja poprzez semafory i pamięć współdzieloną.

• Skalowalność systemu poprzez dynamiczne otwieranie i zamykanie drugiego okienka rejestracji.

**Procedura pacjent.c**

1. **Struktura programu**
   1. Główna funkcja main() przetwarza dane pacjenta przekazane jako argumenty i zarządza procesem pacjenta.
   2. Funkcja pacjent\_zarzadzanie(Pacjent pacjent) odpowiada za wysyłanie pacjenta do odpowiednich kolejek i oczekiwanie na rejestrację.
2. **Funkcja** pacjent\_zarzadzanie()
   1. Zarządza przejściem pacjenta przez kolejkę zewnętrzną do rejestracji.
   2. Tworzy dostęp do kolejki zewnętrznej za pomocą msgget() z kluczem KOLEJKA\_ZEWNETRZNA.
   3. Dodaje pacjenta do kolejki zewnętrznej za pomocą msgsnd() z komunikatem zawierającym dane pacjenta.
   4. Pętla sprawdza, czy przychodnia jest otwarta (czy\_przychodnia\_otwarta()). Jeśli nie, pacjent czeka i próbuje ponownie.
   5. Po otwarciu przychodni pacjent próbuje uzyskać dostęp do kolejki rejestracyjnej za pomocą msgget(KOLEJKA\_REJESTRACJA).
3. **Funkcja** main()
   1. Sprawdza liczbę argumentów wejściowych. Oczekuje sześciu argumentów:
      1. id - unikalny identyfikator pacjenta.
      2. wiek - wiek pacjenta.
      3. priorytet - status VIP (1 - VIP, 0 - zwykły pacjent).
      4. rodzic\_obecny - flaga obecności rodzica (1 - obecny, 0 - brak).
      5. Lekarz - typ lekarza, do którego pacjent chce się zarejestrować.
      6. pid - identyfikator procesu pacjenta.
   2. Wypełnia strukturę Pacjent danymi wejściowymi.
   3. Przekazuje strukturę Pacjent do funkcji pacjent\_zarzadzanie().
4. **Działanie kolejek komunikatów**
   1. Kolejka zewnętrzna (KOLEJKA\_ZEWNETRZNA):
      1. Pacjenci są dodawani do tej kolejki po przybyciu do przychodni.
   2. Kolejka rejestracyjna (KOLEJKA\_REJESTRACJA):
      1. Pacjenci przechodzą z niej do rejestracji.

**Kluczowe funkcjonalności**

* Pacjent próbuje dołączyć do kolejki zewnętrznej i czeka na możliwość rejestracji w przychodni.
* W przypadku zamknięcia przychodni pacjent nie zostaje wpuszczony i ponawia próbę w odpowiednich odstępach czasu.

**Procedura lekarz.c**

1. **Zmienne globalne**
   1. **Flagi zajętości i zakończenia pracy**:
      1. zajety\_POZ[2], zakoncz\_prace\_POZ[2] – status lekarzy POZ (czy zajęci i czy kończą pracę).
      2. zajety\_SPEC[5], zakoncz\_prace\_SPEC[5] – status lekarzy specjalistów.
   2. **Pamięć współdzielona**:
      1. shm\_id – identyfikator pamięci współdzielonej.
      2. liczba\_osob – wskaźnik do liczby osób w przychodni.
   3. suma\_kolejek – wskaźnik do liczby pacjentów w kolejkach.
2. **Obsługa sygnałów**
   1. **Funkcje** signal\_handler\_lekarz\_POZ **i** signal\_handler\_lekarz\_SPEC:
      1. Odbierają sygnał SIGTERM (sygnał zakończenia pracy).
      2. Jeśli lekarz jest **zajęty**, ustawia flagę zakończenia pracy (zakoncz\_prace\_\*).
      3. Jeśli lekarz **nie jest zajęty**, kończy proces natychmiast.
3. **Obsługa lekarza POZ (lekarz\_poz())**
   1. **Kroki działania**:

Ustawia **obsługę sygnałów** (SIGTERM).

Pobiera **dostęp do pamięci współdzielonej**.

Tworzy **kolejki pacjentów**:

Zwykła kolejka POZ (id\_kolejka).

Kolejka VIP (id\_kolejka\_VIP).

Kolejka rejestracyjna (KOLEJKA\_REJESTRACJA).

**Obsługuje pacjentów w kolejności**:

Najpierw pacjenci VIP, potem zwykli pacjenci.

**Losowe kierowanie pacjentów do specjalistów** (20% pacjentów).

**Zakończenie pracy po osiągnięciu limitu pacjentów**.

1. **Obsługa lekarza specjalisty (lekarz\_specjalista())**
   1. **Kroki działania**:

Ustawia **obsługę sygnałów** (SIGTERM).

Pobiera **dostęp do pamięci współdzielonej**.

Tworzy **kolejki pacjentów**:

Zwykła kolejka specjalisty (id\_kolejka).

Kolejka VIP (id\_kolejka\_VIP).

Kolejki badań ambulatoryjnych (id\_kolejka\_badania).

**Obsługuje pacjentów w kolejności**:

Najpierw pacjenci z badań ambulatoryjnych.

Następnie pacjenci VIP.

Na końcu pacjenci z normalnej kolejki.

**Losowe kierowanie pacjentów na badania ambulatoryjne** (10% pacjentów).

**Zakończenie pracy po osiągnięciu limitu pacjentów**.

1. **Kluczowe funkcjonalności**
   1. **Lekarze POZ** obsługują pacjentów, mogą kierować do specjalistów.
   2. **Lekarze specjaliści** obsługują pacjentów, mogą kierować na badania ambulatoryjne.
   3. **System dynamicznie zarządza kolejkami pacjentów**.
   4. **Lekarze kończą pracę po osiągnięciu limitu pacjentów lub na sygnał** SIGTERM.

**Linki do kluczowych elementów:**

**Main.c**

[**https://github.com/frycu07/Systemy\_Operacyjne\_Projekt/blob/main/main.c**](https://github.com/frycu07/Systemy_Operacyjne_Projekt/blob/main/main.c)

|  |  |
| --- | --- |
| **Funkcja** | **Użyte wywołania systemowe** |
| main() | fork(), execl(), exit(), waitpid(), signal(), pthread\_create(), pthread\_join(), semget(), semctl(), semop(), shmget(), shmat(), shmctl() |
| create\_patient() | fork(), execv(), exit() |
| zakonczenie\_procesow() | waitpid() |
| cleanup\_on\_exit() | wait(), exit(), shmctl(), semctl() |
| signal\_handler() | getpid(), signal(), exit() |
| sigchld\_handler() | waitpid() |
| process\_cleaner() | waitpid() |
| pamiec\_wspoldzielona() | shmget(), shmat() |
| pamiec\_wspoldzielona2() | shmget(), shmat() |

**Rejestracja.c**

[**https://github.com/frycu07/Systemy\_Operacyjne\_Projekt/blob/main/rejestracja.c**](https://github.com/frycu07/Systemy_Operacyjne_Projekt/blob/main/rejestracja.c)

|  |  |
| --- | --- |
| **Wystąpienia** | **Funkcje** |
| main() | fork(), pthread\_create(), pthread\_join(), signal(), exit() |
| create\_osoby\_do\_przyjecia\_lekarz() | shmget(), shmat(), semget(), semctl(), exit() |
| remove\_osoby\_do\_przyjecia\_lekarz() | shmctl(), semctl() |
| uzyskaj\_pamiec\_wspoldzielona() | shmget(), shmat() |
| pamiec\_wspoldzielona2() | shmget(), shmat() |
| zakonczenie\_procesow() | waitpid() |
| blokuj\_osoby\_do\_przyjecia\_lekarz() | zmniejsz\_semafor() |
| odblokuj\_osoby\_do\_przyjecia\_lekarz() | zwieksz\_semafor() |
| rejestracja() | signal(), msgget(), msgsnd(), msgrcv(), exit() |
| zarzadz\_kolejka\_zewnetrzna() | msgget(), msgsnd(), msgrcv(), shmget(), shmat() |
| zarzadz\_i\_monitoruj\_rejestracje() | fork(), msgctl(), kill(), zmniejsz\_semafor(), zwieksz\_semafor() |
| zapisz\_do\_raportu() | fopen(), fprintf(), fclose() |

**Lekarz.c**

[**https://github.com/frycu07/Systemy\_Operacyjne\_Projekt/blob/main/lekarz.c**](https://github.com/frycu07/Systemy_Operacyjne_Projekt/blob/main/lekarz.c)

|  |  |
| --- | --- |
| **Wystąpienia** | **Funkcje** |
| main() | atoi(), fprintf(), exit() |
| signal\_handler\_lekarz\_POZ() | signal(), exit() |
| signal\_handler\_lekarz\_SPEC() | signal(), exit() |
| uzyskaj\_pamiec\_wspoldzielona() | shmget(), shmat() |
| badania() | msgget(), msgrcv(), msgsnd(), perror(), exit() |
| lekarz\_poz() | signal(), msgget(), msgrcv(), msgsnd(), kill(), waitpid(), exit() |
| lekarz\_specjalista() | signal(), msgget(), msgrcv(), msgsnd(), kill(), waitpid(), exit() |

**Pacjent.c**

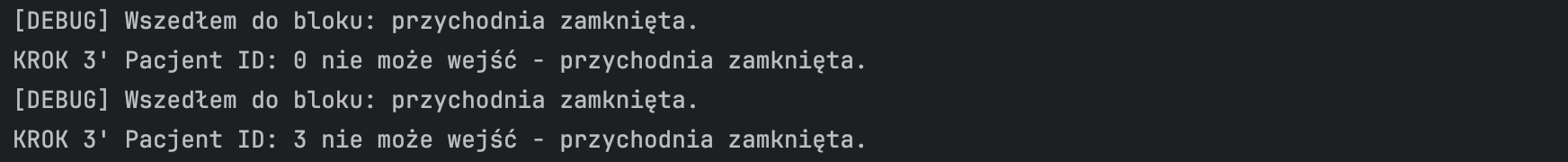
[**https://github.com/frycu07/Systemy\_Operacyjne\_Projekt/blob/main/pacjent.c**](https://github.com/frycu07/Systemy_Operacyjne_Projekt/blob/main/pacjent.c)

|  |  |
| --- | --- |
| main() | signal(), atoi(), fprintf(), exit() |
| pacjent\_zarzadzanie() | msgget(), msgsnd(), perror(), exit(), sleep() |
| signal\_handler\_pacjent() | getpid(), printf(), exit() |

**TESTY**

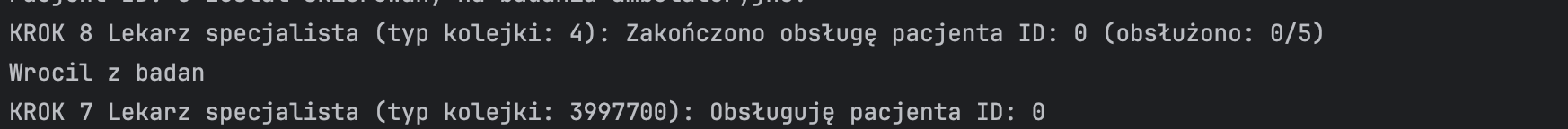
**TEST 1**

* **Cel testu:**
  + Weryfikacja czy po godzinach przyjęć pacjent może wejść do przychodni
* **Szczegóły:**
  + Symulacja:
    1. Godzina zamknięcia w czas.c ustawiona na wcześniejszą niż obecna
  + Oczekiwane wyniki:
    1. Pacjent nie będzie mógł wejść do przychodni z kolejki zewnetrznej
* **Rezultat:**
  + Pozytywny

****

**TEST 2**

* **Cel testu:**
  + Weryfikacja czy po wysłaniu na badania ambulatoryjne pacjent wraca do lekarza specjalisty
* **Szczegóły:**
  + Symulacja:
    1. Losowy pacjent (10%)
  + Oczekiwane wyniki:
    1. Pacjent wróci do tego samego lekarza po badaniach.
* **Rezultat:**
  + Pozytywny

****

**TEST 3**

* **Cel testu:**
  + Weryfikacja czy otworzy się drugie okienko rejestracji, jeśli liczba pacjentów w kolejce do rejestracji przekroczy polowe pojemności przychodni.
* **Szczegóły:**
  + Symulacja:
    1. MAX liczba osób w przychodni 20
  + Oczekiwane wyniki:
    1. Przy 11 osobach w kolejce otworzy się drugie okienko (pierwsze ma nazwę rejestracja 0)
* **Rezultat:**
  + Pozytywny

**Obraz zawierający tekst, zrzut ekranu, Czcionka

Opis wygenerowany automatycznie**

**TEST 4**

* **Cel testu:**
  + Weryfikacja czy jeżeli lekarz obsłuży wymaganą liczbę pacjentów to zakończy pracę.
* **Szczegóły:**
  + Symulacja:
    1. MAX liczba osób dla lekarza specjalisty 1 to 5 osob
  + Oczekiwane wyniki:
    1. Lekarz nie będzie przyjmował więcej pacjentów i zakończy pracę
* **Rezultat:**
  + Pozytywny

**Obraz zawierający tekst, zrzut ekranu, Czcionka, czarne

Opis wygenerowany automatycznie**