Zadanie 1.2.
Algorytm FIFO chyba każdy wie jak działa.
Algorytm RR (round robin) działa jak wywłaszczanie w procesów przez system operacyjny. Zakładam, że jeśli zadanie skończy się przed upływem kwantu czasu, to kolejne zadanie dostaje nowy kwant czasu - czyli 2s.

$d_n = t_n^- - t_n^+$

ale wszystkie żądania przychodzą w tym samym momencie

$t_1^+ = 0 \quad t_2^+ = 0 \quad t_3^+ = 0$

więc każde opóźnienie jest równe momentowi zakończenia obsługi

$d_n = t_n^-$

Średnie opóźnienie: $d_m = \frac{\sum(d_n)}{n}$

a) XYZ

FIFO: $t_1^- = 7s \quad t_2^- = 8s \quad t_3^- 11s \quad d_m = \frac{26}{3} = 8\frac{2}{3}s$

RR: $t_1^- = 11s \quad t_2^- = 3s \quad t_3^- 8s \quad d_m = \frac{22}{3} = 7\frac{1}{3}s$

b) YZX

FIFO: $t_1^- = 1s \quad t_2^- = 4s \quad t_3^- 11s \quad d_m = \frac{16}{3} = 5\frac{1}{3}s$

RR: $t_1^- = 1s \quad t_2^- = 6s \quad t_3^- 11s \quad d_m = \frac{18}{3} = 6s$

c) XZY

FIFO: $t_1^- = 7s \quad t_2^- = 10s \quad t_3^- 11s \quad d_m = \frac{28}{3} = 9\frac{1}{3}s$

RR: $t_1^- = 11s \quad t_2^- = 8s \quad t_3^- 5s \quad d_m = \frac{24}{3} = 8s$

Widać, że stosując metodę RR, mniejsze zadania muszą czekać stosunkowo krócej niż większe. Także średnie opóźnienie jest zawsze mniejsze niż w FIFO.

edit:
dopiero zauważyłem, że w b) średnie opóźnienie jest mniejsze dla FIFO

Zad 1.3

$b_{śr}$ = 5000 operacji

$a_{śr}$ = 1s/800transakcji

υ = 4 000 000 operacji/s

Prawo Little'a:

$N_{śr} = \frac{1-L}{a_{śr}} d_{śr}$

W zadaniu nie ma mowy o skończonej długości bufora, w związku z tym frakcja utraconych transakcji = 0 (L = 0). Pozostaje wyliczyć średnie opóźnienie systemowe (czas pozostawania zadania w systemie).

$$d_{śr} = \frac{b_{śr}}{\upsilon} = \frac{5000}{4000000} = \frac{1}{800}$$

Pozostaje podstawić.

$$N_{śr} = \frac{1-0}{\frac{1}{800}} \frac{1}{800} = 1$$

Odpowiedź: Średnio w systemie znajduje się 1 transakcja jednocześnie.

Zad 2.2

S = 1
Q < ∞
r > 1

Q zmierza do nieskończoności, procesor się nie wyrabia (r > 1), można więc przyjąć, że współczynnik bezczynności procesora (p_0) wynosi 0.
Równanie ciągłości przepływu:

$$1 - p_0 = (1 - L)r$$

Podstawiamy i wyliczamy L.

$$1 - 0 = (1 - L)r$$

$$\frac{1}{r} = 1 - L$$

$$L = 1 - \frac{1}{r}$$

$$L > 0$$

Ponieważ r > 1, łatwo zauważyć, że L nigdy nie będzie równe 0. Oznacza to, że frakcja utraconych pakietów może się zbliżyć bardzo blisko do 0 (r musi być niewiele większe od 1), ale samego zera nigdy nie osiągnie.

Practice Set 3 / Problem 1

Ilość źródeł J=50

Think time = $\frac{2}{3}$ s = mean system interval

Am= $\frac{2}{3}$ s

Bm=80%*1000*8+20%*160*8=6656 bits

v = 1 Mbit/s

p0=prawdopodobieństwo bezczynności = 75%

τm=bm/v=0,006656 s

rownanie ciągłości przepływu

$$1 - p0 = J * \left(\frac{1-L}{am}\right) * \tau m$$

$$L = 1 - \frac{(1-p0)*am}{J*\tau m}$$

L=1- $\frac{0{,}25*\left(\frac{2}{3}\right)}{50*0{,}006656}$

L=0,4992

Dans w zadaniu 3.2

r = 0.75 ale p0 nie mo¿emy sobie tak wydedukowaæ, ¿e to 0.25
(niestety nie wiem dlaczego, mo¿e ktoœ wie?)

Trzeba policzyæ równanie z:
p0 + p1 + p2 = 1 // bo mamy tylko 3 mozliwosci
1 - p0 = (1-L)r // gdzie r = 0.75 a L = p2
p0 >= p1 >= p2

Wynik wychodzi
4/10 >= p1 >= 3/11

Prz czym:
J=N, bo końcówka generuje zg³oszenie po przetworzeniu poprzedniego, dlatego w systemie jest tyle zg³oszeñ ile koñcówek.
Prawo Little'a w postaci:
J=N=(1-L)(w+tau+h)/a
gdzie L=0, bo nie am mowy o utraconych,
d=(w+tau+h), bo mamy opóŸnienie h, które wprowadzaj¹ koñcówki, opóŸnienie buforowania w i czas obs³ugi tau (=b/v)
oraz cyrkulacja w ca³ym systemie:
a = (1-p0)/tau, bo przetwarzamy w systemie z wydajnoœci¹ tau (czyli interwa³ 1/tau), ale tylko wtedy, kiedy mamy zg³oszenia, st¹d mamy (1-p0)

Mam nadziejê, ¿e nic nie pomiesza³em, ale mniej wiêcej, tak go zrozumia³em.
Z tym, ¿e nie wiem, czy w a (1-p0) nie powinno byæ w mianowniku, bo napisa³ to, tak, ¿e do koñca nie wiem.