1. Które z wymienionych informacji zawiera tablica ramek?  
 [wielokrotnego]

* Strony procesu
* Adresy ramek przydzielonych procesowi
* Informacje o wolnych ramkach pamięci operacyjnej

2. Które z wymienionych informacji są zawarte w pliku typu katalog?  
[wielokrotnego]

* numery i-węzłów plików w tym katalogu
* informacje zawarte w plikach w tym katalogu
* nazwa tego katalogu
* nazwy plików w tym katalogu

3. Która funkcja systemowa tworzy nowy proces

[jednokrotnego]

* exec
* write
* open
* fork

4. Które z wymienionych informacji są w tablicy stron?  
[wielokrotnego]

* Adres kolejnej wolnej ramki
* Bit poprawności (Magda note)
* Bit ochrony tej strony
* Adres bazowy ramki, w której jest ta strona w pamięci

5. Który sposób przydziału miejsca na dysku rozwiązuje problem zewnętrznej fragmentacji wolnej przestrzeni na dysku?  
[wielokrotnego]

* Przydział listowy
* Przydział indeksowy
* Przydział ciągły

6. Który z wymienionych struktur systemu UNIX znajduje informację jaki podprogram obsługi urządzeń należy uruchomić w celu wykonania określonej operacji na konkretnym urządzeniu? [jednokrotnego]

* Plik specjalny
* Tablica rozdzielcza urządzeń
* W żadnej z wymienionych
* i-węzeł
* Blok kontrolny procesu

7. Który z wymienionych sposobów przydziału miejsca na dysku umożliwia bezpośredni dostęp do informacji pliku? [wielokrotnego]

* Przydział ciągły
* Przydział listowy
* Przydział indeksowy

8. Co jest przedmiotem standaryzacji systemów otwartych? [jednokrotnego]

* Podprogramy obsługi urządzeń
* Oprogramowanie Jądra systemu operacyjnego
* Rozwiązania sprzętowe (hardware)
* Zestaw funkcji systemowych - odwołań systemowych

9. W której z wymienionych struktur system UNIX znajduje informację dotyczącą typu urządzenia w celu wykonania określone operacji na tym urządzeniu? [jednokrotnego]

* Tablica rozdzielcza
* Plik specjalny
* i-węzeł
* Katalog

10. W systemie plików blok na rozmiar 16kB i stosowane jest adresowanie 32 bitowe. Ile wskaźników adresowych zawiera jeden blok adresowania pośredniego? [jednokrotnego]

* 512
* 4096
* 1024
* 2048

11. Dla istniejącego już procesu wywołano funkcję systemową exec.

Które z podanych informacji zmieniane są w przestrzeni pamięci operacyjnej przedzielonej temu procesowi? [wielokrotnego]

* Zmienne lokalne
* Kod programu
* Informacje o właścicielu i grupie
* Informacja o bieżącym katalogu

12. W kolejce procesów gotowych znajdują się procesy:

1. Wykonywany w trybie użytkownika czekający 0,4 sekundy
2. Wykonywany w trybie jądra oczekujący na i-węzeł, czekający w kolejce 0,2 sekundy
3. Wykonywany w trybie jądra oczekujący na zakończenie potomka czekający w kolejce 0,1 sekundy

Jaka będzie kolejność przydziału procesora? [wielokrotnego]

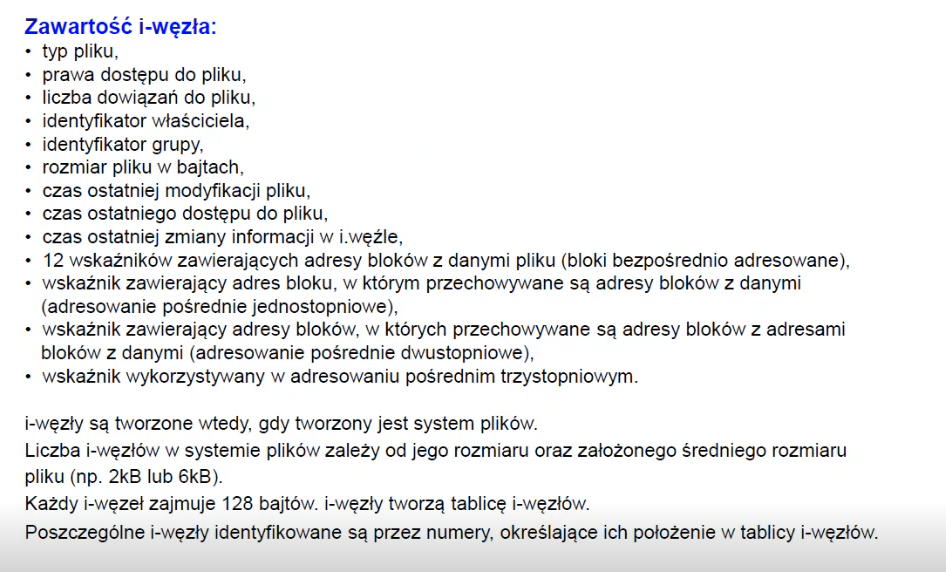
* C, B. A
* B, C, A
* C, A, B
* A, B, C

13. Które z podanych informacji zawiera blok kontrolny procesu? [wielokrotnego]

* Stan licznika rozkazów
* Kod programu wykonywanego przez proces
* Informacje dotyczące zarządzania pamięcią
* Informacje o stanie operacji wejścia wyjścia
* Informacje zawarte w rejestrach procesora
* Wywołania systemowe

14. Które z wymienionych informacji są zawarte w i-węźle? [wielokrotnego]

* wskaźniki adresowe bloków pliku na dysku
* nazwa pliku
* prawa dostępu do pliku
* rozmiar systemu plików - Magda by nie zaznaczyła
* wskaźniki adresowe do wolnych bloków na dysku
* czas modyfikacji pliku



///////////////////////////////////////////////////////////////////////////////  
Część otwarta:  
///////////////////////////////////////////////////////////////////////////////

**1.1) O godzinie 20.00 utworzono nowy plik wydając następujące polecania:**

**$date > /tmp/plikA ; ln /tmp/plikaA /tmp/plikB**

**o godz. 21:30 wydano polecenie:**

**$date >> /tmp/plikA; ls -l /tmp/plikB**

**następnie o godz 22:30 wydano polecenie:**

**$ls -l /tmp/plikA**

**Podaj dokładnie, co znajduje się w i-węźle pliku /tmp/plikA (Uwaga: przyjąć że wyjście polecenia date zajmuje 30 bajtów). Odpowiedź uzasadnić.**

W zadaniu jest “co znajduje się w plikA”, ale jest **ln** to wiadomo, ze to dotyczy tego samego obiektu, także nawet jak byłby plikB to odpowiedzi byłyby te same

**zawartość i-wezła:**

**typ pliku**: plik zwykły

**prawa dostępu do pliku**: r w - r w - r- -

**liczba dowiązań do pliku:** 2

**identyfikator właściciela:** user\_id (jakis ogólny podać albo np swój: kowalski czy coś)

**identyfikator grupy:** id\_group (tez ogólnie np. users domain)

**rozmiar pliku w bajtach:** 60 (bo 2x zapisano *date* w pliku a jeden *date* zajmuje 30)

**czas ostatniej modyfikacji pliku:** 21:30 (bo wtedy dopisano do pliku drugie *date*)

**czas ostatniego dostępu do pliku:** 20:00 - bo ani *date* ani *ls* nie zmieniają tu nic, jakby było *cat* to co innego - sprawdzone w putty :B także tutaj ostatni dostęp=utworzenie

**czas ostatniej zmiany informacji w i-węźle:** 21:30 bo wtedy zmieniono plik (dopisano drugie *date*) i przez to zmienił się czas ostatniej modyfikacji pliku

TO SAMO ZADANIE **1**, INNE DANE!!!!

**1.2) O godzinie 14.00 utworzono nowy plik wydając następujące polecenia:**

$ date > /tmp/plikA ; ln /tmp/plikA /tmp/plikC  
następnie o godz. 16.00 wydano polecenie  
 $ date >> /tmp/plikA; cat /tmp/plikC  
a o godz. 17.30 wydano polecenie  
 $ ln /tmp/plikA /tmp/plikB

Podaj dokładnie, co znajduje się w i-węźle pliku /tmp/plikC (Uwaga: przyjąć że wyjście polecenia date zajmuje 30 bajtów). Odpowiedź uzasadnić.

**typ pliku**: typ dowiązania

**uprawnienia:** (zależą od systemu i umask, np u mnie na malinie: -rw -r- -r- )

**ilość dowiązań:** 3

**autor:** twójLoginDoUbi (podmieńcie to parówy)

**grupaAutora:** users

**waga:** 60 bajtów (2x date użycie, 30\*2 = 60)

**czas ostatniej modyfikacji pliku**: 16:00

**czas ostatniego dostępu do pliku:** 16:00

**czas ostatniej zmiany informacji w i-węźle**: 17.30

TO SAMO ZADANIE **1**, INNE DANE!!!!

**1.3) O godzinie 16.30 utworzono nowy plik wydając następujące polecenia:**

$ date > /tmp/plikA ;  
następnie o godz. 17.30 wydano polecenie  
 $ ln /tmp/plikA /tmp/plikB; cat /tmp/plikA  
a o godz.17.45 wydano polecenie  
 $ ln /tmp/plikA /tmp/plikC

Podaj dokładnie, co znajduje się w i-węźle pliku /tmp/plikC (Uwaga: przyjąć że wyjście polecenia date zajmuje 30 bajtów). Odpowiedź uzasadnić.

**typ pliku**: typ dowiązania

**uprawnienia:** (zależą od systemu i umask, np u mnie na malinie: -rw-r--r–)

**ilość dowiązań:** 3

**autor:** twójLoginDoUbi(podmieńcie to parówy)

**grupaAutora:** users

**waga:** 30 bajtów (1x użycie date, 1x30bajtów)

**czas ostatniej modyfikacji pliku**: 16:30

**czas ostatniego dostępu do pliku:** 17:30

**czas ostatniej zmiany informacji w i-węźle**: 17.45

TO SAMO ZADANIE **1**, INNE DANE!!!!

**1.4) O godzinie 21.10 utworzono nowy plik wydając następujące polecenia:**

$ date > /tmp/plikA ; cat /tmp/plikA  
następnie o godz. 21.30 wydano polecenie  
 $ ln /tmp/plikA /tmp/plikC;   
a o godz. 22.20 wydano polecenie  
 $ ln /tmp/plikA /tmp/plikB; date >> /tmp/plikC

Podaj dokładnie, co znajduje się w i-węźle pliku /tmp/plikB (Uwaga: przyjąć że wyjście polecenia date zajmuje 30 bajtów). Odpowiedź uzasadnić.

**typ pliku**: typ dowiązania

**uprawnienia:** (zależą od systemu i umask, np u mnie na malinie: -rw-r--r–)

**ilość dowiązań:** 3

**autor:** twójLoginDoUbi(podmieńcie to parówy)

**grupaAutora:** users

**waga:** 60 bajtów

**czas ostatniej modyfikacji pliku**: 22.20

**czas ostatniego dostępu do pliku:** 21.110

**czas ostatniej zmiany informacji w i-węźle**: 22.20

**2.1) Dana jest mapa bitowa systemu plików, w którym blok ma 8kB, a fragment 2kB. Poniżej mapy podane są adresy początkowe kolejnych fragmentów. Należy zaadresować plik o rozmiarze 27kB. Należy podać (w wolnym wierszy tabeli) nowy stan mapy bitowej, a także ile i które wskaźniki adresowe i-węzła będą wykorzystane do zaadresowania pliku i jakie adresy będą w nich umieszczone? Odpowiedź uzasadnić.  
  
mapa bitowa:**

**1 0 1 0 1 0 1 1 0 0 0 0 1 0 0 1 0 0 0 0 0 0 0 0**

**adresy:**

**0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23**

<https://ch3shiredev.github.io/systemy-operacyjne-solver/#/adressing>

Rozmiar pliku: 27kB **(Przepisujemy z polecenia)**

Liczba Bloków: 3 x 8Kb = 24kB **( sprawdzamy ile bloków o rozmiarze 8kB zmieścimy w rozmiarze pliku** (24kB z 27kB)

Liczba fragmentów: 2 x 2kB = 4kB **(sprawdzamy ile fragmentów o rozmiarze 2kB zmieścimy w reszcie miejsca wolnego pliku czyli 27-24=3kB, musimy zająć 4 kB)**

Miejsce zajęte na dysku: 24kB + 4kB = 28kB **( Sumujemy nasze fragmenty i bloki)**

**Stara Mapa bitowa:**

| Mapa bitowa | 1 0 1 0 | 1 0 1 1 | 0 0 0 0 | 1 0 0 1 | 0 0 0 0 | 0 0 0 0 |
| --- | --- | --- | --- | --- | --- | --- |
| Adresy: | 0 1 2 3 | 4 5 6 7 | 8 9 10 11 | 12 13 14 15 | 16 17 18 19 | 20 21 22 23 |

**Nowa mapa bitowa:**

| Mapa bitowa | 1 0 1 0 | 1 0 1 1 | 1 1 1 1 | 1 1 1 1 | 1 1 1 1 | 1 1 1 1 |
| --- | --- | --- | --- | --- | --- | --- |
| Adresy: | 0 1 2 3 | 4 5 6 7 | 8 9 10 11 | 12 13 14 15 | 16 17 18 19 | 20 21 22 23 |

| **Adresy w i-węźle** | |
| --- | --- |
| **Nr Pozycji** | **Adres** |
| 1 | 8 |
| 2 | 16 |
| 3 | 20 |
| 4 | 13 |
| 5 | Wpisanie tu 14 to błąd (fragmenty obok siebie) |
| … |  |
| 15 |  |

**Miejsce na dysku stracone w wyniku wewnętrznej fragmentacji: 1kB** (Sprawdzamy od lewej do prawej gdzie mamy najbliższe wolne całe bloki czyli cztery zera, w tym przypadku będą to adresy początkowe 8,16,20 . Następnie szukamy najbliższego wolnego fragmentu, czyli adres początkowy 13)

TO SAMO ZADANIE **2**, INNE DANE!!!!

**2.1) Dana jest mapa bitowa systemu plików, w którym blok ma 8kB, a fragment 2kB. Poniżej mapy podane są adresy początkowe kolejnych fragmentów. Należy zaadresować plik o rozmiarze 29kB. Należy podać (w wolnym wierszy tabeli) nowy stan mapy bitowej, a także ile i które wskaźniki adresowe i-węzła będą wykorzystane do zaadresowania pliku i jakie adresy będą w nich umieszczone? Odpowiedź uzasadnić.  
  
mapa bitowa:**

**1 0 0 0 1 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0**

**adresy:**

**0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23**

rozmiar 29 kb

blok 8 kb

fragment 2kB

ilość bloków 3 (3\*8kB = 24kB, mniej niż 29 kB, ale nie chcemy więcej)

ilość fragmentów 3 ( (29 - 24) = 5 kB zostało do pokrycia fragmentami także 3\*2kB, 1 kB straty )

pozycje - adresy:

1 - 8

2 - 12

3 - 16

4 - 1

**Stara Mapa bitowa:**

| Mapa bitowa | 1 0 0 0 | 1 0 0 1 | 0 0 0 0 | 0 0 0 0 | 0 0 0 0 | 0 1 0 0 |
| --- | --- | --- | --- | --- | --- | --- |
| Adresy: | 0 1 2 3 | 4 5 6 7 | 8 9 10 11 | 12 13 14 15 | 16 17 18 19 | 20 21 22 23 |

**Nowa mapa bitowa:**

| Mapa bitowa | 1 1 1 1 | 1 0 0 1 | 1 1 1 1 | 1 1 1 1 | 1 1 1 1 | 0 1 0 0 |
| --- | --- | --- | --- | --- | --- | --- |
| Adresy: | 0 1 2 3 | 4 5 6 7 | 8 9 10 11 | 12 13 14 15 | 16 17 18 19 | 20 21 22 23 |

miejsce zajęte na dysku: 24kB + 6kB = 30 kB, strata 1kB przy adresowaniu

| **Adresy w i-węźle** | |
| --- | --- |
| **Nr Pozycji** | **Adres** |
| 1 | 8 |
| 2 | 12 |
| 3 | 16 |
| 4 | 1 |
| 5 | wpisz tu 2 to wjeb |
| … | wpisz tu 3 to wjeb |
| 15 |  |

TO SAMO ZADANIE **2**, INNE DANE!!!!

**2.2) Dana jest mapa bitowa systemu plików, w którym blok ma 4kB, a fragment 1kB. Poniżej mapy podane są adresy początkowe kolejnych fragmentów. Należy zaadresować plik o rozmiarze 14kB. Należy podać (w wolnym wierszy tabeli) nowy stan mapy bitowej, a także ile i które wskaźniki adresowe i-węzła będą wykorzystane do zaadresowania pliku i jakie adresy będą w nich umieszczone? Odpowiedź uzasadnić.  
  
mapa bitowa:**

**0 1 0 1 1 0 0 1 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0**

**adresy:**

**0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23**

rozmiar pliku 14 kB

blok 4 kB

fragment 1kB

ilość bloków 3 (3\*4kB = 12kB, mniej niż 14kB, ale nie chcemy więcej)

ilość fragmentów 2 ( (14 - 12) = 2 kB zostało do pokrycia fragmentami także 2 \* 1kB, 0 kB straty )

**Stara Mapa bitowa:**

| Mapa bitowa | 0 1 0 1 | 1 0 0 1 | 0 0 0 0 | 0 1 0 0 | 0 0 0 0 | 0 0 0 0 |
| --- | --- | --- | --- | --- | --- | --- |
| Adresy: | 0 1 2 3 | 4 5 6 7 | 8 9 10 11 | 12 13 14 15 | 16 17 18 19 | 20 21 22 23 |

**Nowa mapa bitowa:**

| Mapa bitowa | 0 1 0 1 | 1 0 0 1 | 1 1 1 1 | 0 1 1 1 | 1 1 1 1 | 1 1 1 1 |
| --- | --- | --- | --- | --- | --- | --- |
| Adresy: | 0 1 2 3 | 4 5 6 7 | 8 9 10 11 | 12 13 14 15 | 16 17 18 19 | 20 21 22 23 |

| **Adresy w i-węźle** | |
| --- | --- |
| **Nr Pozycji** | **Adres** |
| 1 | 8 |
| 2 | 16 |
| 3 | 20 |
| 4 | 14 |
| 5 | >nic< |
| … | >nic< |
| 15 | >nic< |

**3.1)** W chwili **0 sekund** zostaje zgłoszone zadanie użytkownika i utworzony proces **1**,   
**10 sekund później** zostaje utworzony proces **2** i jeszcze **25 sek. później** po procesie 2,  
proces **3**. Przewidywany czas wykonania procesu **1 wynosi 30 sek**., procesu **2 wynosi 60 sek**.,   
a procesu **3 wynosi 20 sek**. Procesor jest dostępny do przetwarzania tych procesów od chwili   
**15-ej sekundy.**   
Czas przełączania kontekstu proszę pominąć.

Policzyć średni czas przetwarzania tych procesów dla algorytmu SJF bez wywłaszczania. Należy podać sposób rozwiązania i uzasadnienie. Proszę na wykresie czasu procesora zaznaczyć okresy wykonywania tych procesów.

Trzy procesy o czasach wykonywania: 1 - 30s, 2 - 60s, 3 - 20s.

Zgłoszenia procesów:

1 2 3

|| || || czasy zgłoszenia procesów

0s 10s 35s

czasy wykonywania

| procesor zajęty | proces 1 (p1) | proces 3 (p3) | proces 2 (p2) |
| --- | --- | --- | --- |

0 15s 45s 65s 125s

start procesor zajęty koniec p1 koniec p3 koniec p2

Czasy przetwarzania: (czas końca - czas zgłoszenia)

Δt1 = 45s - 0s = 45s

Δt2 = 125s - 10s = 115s

Δt3 = 65s - 35s = 30s

średni czas przetwarzania: (45s + 115s + 30s) / 3 = 63,(3)s

TO SAMO ZADANIE **3**, INNE DANE!!!!

**3.2)** W chwili **0 sekund** zostaje zgłoszone zadanie użytkownika i utworzony proces **1**,   
**15 sekund później** zostaje utworzony proces **2** i jeszcze **30 sek.** później po procesie 2,   
- proces **3**. Przewidywany czas wykonania procesu **1 wynosi 40 sek.**, procesu **2 wynosi 60 sek**.,   
a procesu **3 wynosi 20 sek**. Procesor jest dostępny do przetwarzania tych procesów od chwili **25-ej sekundy**.   
Czas przełączania kontekstu proszę pominąć.  
  
Policzyć średni czas przetwarzania tych procesów dla algorytmu SJF bez wywłaszczania. Należy podać sposób rozwiązania i uzasadnienie. Proszę na wykresie czasu procesora zaznaczyć okresy wykonywania tych procesów.

Trzy procesy o czasach wykonywania: 1 - 45s, 2 - 60s, 3 - 20s.

Zgłoszenia procesów:

1 2 3

|| || || czasy zgłoszenia procesów

0s 15s 45s

czasy wykonywania

| procesor zajęty | proces 1 (p1) | proces 3 (p3) | proces 2 (p2) |
| --- | --- | --- | --- |

0 25s 70s 90s 150s

start procesor zajęty koniec p1 koniec p3 koniec p2

Czasy przetwarzania:

Δt1 = 70s - 0s = 70s

Δt2 = 150s - 15s = 135s

Δt3 = 90s - 45s = 45s

średni czas przetwarzania: (70s+135s+45s) / 3 = 83,(3)s

TO SAMO ZADANIE **3**, INNE DANE!!!!

**3.3)** W chwili **0 sekund** zostaje zgłoszone zadanie użytkownika i utworzony proces **1**,   
**cztery sekund później** zostaje utworzony proces **2** i jeszcze **15 sek.** później po procesie 2,   
- proces **3**. Przewidywany czas wykonania procesu **1 wynosi 30 sek.**, procesu **2 wynosi 60 sek**.,   
a procesu **3 wynosi 10 sek**. Procesor jest dostępny do przetwarzania tych procesów od chwili **15-ej sekundy**.   
Czas przełączania kontekstu proszę pominąć.  
  
Policzyć średni czas przetwarzania tych procesów dla **algorytmu algorytmu rotacyjnego, gdy kwant czasu wynosi 10 sek**. Należy podać sposób rozwiązania i uzasadnienie. Proszę na wykresie czasu procesora zaznaczyć okresy wykonywania tych procesów.

Kwant: 10s

Trzy procesy o czasach wykonywania: 1 - 30s, 2 - 60s, 3 - 10s

Zgłoszenia procesów:

1 2 3

|| || || czasy zgłoszenia procesów

0s 4s 19s

p1 - proces 1

p2 - proces 2

p3 - proces 3

czasy wykonywania:

| procesor zajęty | p1 | p2 | p3 | p1 | p2 | p1 | p2 | p2 | p2 | p2 |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |

0 15 25 35 45 55 65 75 85 95 105 115

start koniec p3 koniec p1 koniec p2

Czasy przetwarzania: (czas końca - czas zgłoszenia)

Δt1 = 75 - 0s = 75s

Δt2 = 115s - 4s = 111s

Δt3 = 45s - 19s = 26s

średni czas przetwarzania: (75s + 111s + 26s) / 3 = 60,(6)s

**4.1)** W trakcie wykonywania pewnego 900-bajtowego programu zaobserwowano następujący ciąg odwołań do pamięci (podane liczby wskazują adresy komórek pamięci - nie są to numery stron):

208, 445, 173, 835, 620, 231, 890, 509, 176, 234, 630, 185, 210, 651, 31, 165, 856, 434, 265, 153, 46, 858,

Wyznacz ciąg odwołań do stron, zakładając, że rozmiar strony wynosi **200 bajtów**, Następnie, zakładając, że programowi temu przydzielono w pamięci **600 bajtów**, wyznacz liczby błędów braku stron dla algorytmu **OPTYMALNEGO**. Należy podać sposób rozwiązania i uzasadnienie. Przedstawić zmiany zawartości ramek przydzielonych procesowi.

ilość ramek = 600 bajtów / 200 bajtów = 3

każda strona = 200 bajtów, cały program ma 900, także

strona 0: 0-199 bajtów

strona 1: 200-399 bajtów

strona 2: 400-599 bajtów

strona 3: 600-799 bajtów

strona 4: 800-999 bajtów

przykład przeliczenia adresu na stronę: 410 jest w zasięgu 400-599, zatem 2 strona etc.

| komórki | 208 | 445 | 173 | 835 | 620 | 231 | 890 | 509 | 176 | 234 | 630 | 185 | 210 | 651 | 31 | 165 | 856 | 434 | 265 | 153 | 46 | 858 |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| strony: | 1 | 2 | 0 | 4 | 3 | 1 | 4 | 2 | 0 | 1 | 3 | 0 | 1 | 3 | 0 | 0 | 4 | 2 | 1 | 0 | 0 | 4 |

numery stron: 1, 2, 0, 4, 3, 1, 4, 2, 0, 1, 3, 0, 1, 3, 0, 0, 4, 2, 1, 0, 0, 4

Optymalny (zastąp stronę, która najdłużej nie będzie używana)

|  | 1 | 2 | 0 | 4 | 3 | 1 | 4 | 2 | 0 | 1 | 3 | 0 | 1 | 3 | 0 | 0 | 4 | 2 | 1 | 0 | 0 | 4 |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Ramka 1 | 1 | 1 | 1 | 1 | 1 | - | - | 1 | 1 | - | - | - | - | - | - | - | 1 | 1 | - | - | - | 4 |
| Ramka 2 |  | 2 | 2 | 2 | 3 | - | - | 3 | 3 | - | - | - | - | - | - | - | 4 | 2 | - | - | - | 2 |
| Ramka 3 |  |  | 0 | 4 | 4 | - | - | 2 | 0 | - | - |  | - | - | - | - | 0 | 0 | - | - | - | 0 |

ilość błędów: ilość nie pustych kolumn -> 1

TO SAMO ZADANIE **4**, INNE DANE!!!!

**4.2)** W trakcie wykonywania pewnego 350-bajtowego programu zaobserwowano następujący ciąg odwołań do pamięci (podane liczby wskazują adresy komórek pamięci - nie są to numery stron):

https://docs.google.com/document/d/1bz2fIRv3SDrEfVzulUZiO1szL4CZbKiVCjCUgfN5q0I/edit

Wyznacz ciąg odwołań do stron, zakładając, że rozmiar strony wynosi **50 bajtów**, Następnie, zakładając, że programowi temu przydzielono w pamięci **150 bajtów**, wyznacz liczby błędów braku stron dla algorytmu **LRU**. Należy podać sposób rozwiązania i uzasadnienie. Przedstawić zmiany zawartości ramek przydzielonych procesowi.

każda strona = 50 bajtów, cały program ma 400, także

strona 0: 0-49 bajtów

strona 1: 50-99 bajtów

strona 2: 100-149 bajtów

strona 3: 150-199 bajtów

strona 4: 200-249 bajtów

strona 5: 250-299 bajtów

strona 6: 300-349 bajtów

strona 7: 350-399 bajtów

przykład przeliczenia adresu na stronę: 51 jest w zasięgu 50-99, zatem 1 strona etc.

ilość ramek = 150 / 50 = 3

334, 46, 231, 150, 250, 120, 340, 32, 285, 231, 120, 173, 41, 309, 251, 245, 320, 176, 234, 256, 134, 364

| komórki | 334 | 46 | 231 | 150 | 250 | 120 | 340 | 32 | 285 | 231 | 120 | 173 | 41 | 309 | 251 | 245 | 320 | 176 | 234 | 256 | 134 | 364 |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| strony: | 6 | 0 | 4 | 3 | 5 | 2 | 6 | 0 | 5 | 4 | 2 | 3 | 0 | 6 | 5 | 4 | 6 | 3 | 4 | 5 | 2 | 7 |

numery stron: 6, 0, 4, 3, 5, 2, 6, 0, 5, 4, 2, 3, 0, 6, 5, 4, 6, 3, 4, 5, 2, 7

LRU (zastąp stronę, która nie była używana od najdłuższego czasu)

|  | 6 | 0 | 4 | 3 | 5 | 2 | 6 | 0 | 5 | 4 | 2 | 3 | 0 | 6 | 5 | 4 | 6 | 3 | 4 | 5 | 2 | 7 |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Ramka 1 | 6 | 6 | 6 | 3 | 3 | 3 | 6 | 6 | 6 | 4 | 4 | 4 | 0 | 0 | 0 | 4 | - | 4 | - | 4 | 4 | 7 |
| Ramka 2 |  | 0 | 0 | 0 | 5 | 5 | 5 | 0 | 0 | 0 | 2 | 2 | 2 | 6 | 6 | 6 | - | 6 | - | 5 | 5 | 5 |
| Ramka 3 |  |  | 4 | 4 | 4 | 2 | 2 | 2 | 5 | 5 | 5 | 3 | 3 | 3 | 5 | 5 | - | 3 | - | 3 | 2 | 2 |

ilość błędów: 22 - 2 = 20 (liczba kolumn - liczba kolumn z myślnikami)

TO SAMO ZADANIE **4**, INNE DANE!!!!

**4.3)** W trakcie wykonywania pewnego 700-bajtowego programu zaobserwowano następujący ciąg odwołań do pamięci (podane liczby wskazują adresy komórek pamięci - nie są to numery stron):

410, 245, 173, 450, 120, 231, 610, 309, 176, 234, 430, 185, 210, 551, 31, 245, 556, 434, 565, 650, 46, 458

Wyznacz ciąg odwołań do stron, zakładając, że rozmiar strony wynosi **100 bajtów**, Następnie, zakładając, że programowi temu przydzielono w pamięci **200 bajtów**, wyznacz liczby błędów braku stron dla algorytmu **OPTYMALNEGO**. Należy podać sposób rozwiązania i uzasadnienie. Przedstawić zmiany zawartości ramek przydzielonych procesowi.

ilość ramek = 200 bajtów / 100 bajtów = 2

każda strona = 100 bajtów, cały program ma 700, także

strona 0: 0-99 bajtów

strona 1: 100-199 bajtów

strona 2: 200-299 bajtów

strona 3: 300-399 bajtów

strona 4: 400-499 bajtów

strona 5: 500-599 bajtów

strona 6: 600-699 bajtów

przykład przeliczenia adresu na stronę: 410 jest w zasięgu 400-499, zatem 4 strona etc.

410, 245, 173, 450, 120, 231, 610, 309, 176, 234, 430, 185, 210, 551, 31, 245, 556, 434, 565, 650, 46, 458

| komórki | 410 | 245 | 173 | 450 | 120 | 231 | 610 | 309 | 176 | 234 | 430 | 185 | 210 | 551 | 31 | 245 | 556 | 434 | 565 | 650 | 46 | 458 |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| strony: | 4 | 2 | 1 | 4 | 1 | 2 | 6 | 3 | 1 | 2 | 4 | 1 | 2 | 5 | 0 | 2 | 5 | 4 | 5 | 6 | 0 | 4 |

numery stron: 4 2 1 4 1 2 6 3 1 2 4 1 2 5 0 2 5 4 5 6 0 4

Optymalny (zastąp stronę, która najdłużej nie będzie używana)

|  | 4 | 2 | 1 | 4 | 1 | 2 | 6 | 3 | 1 | 2 | 4 | 1 | 2 | 5 | 0 | 2 | 5 | 4 | 5 | 6 | 0 | 4 |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Ramka 1 | 4 | 4 | 4 | - | - | 2 | 6 | 3 | - | 2 | 4 | - | 4 | 5 | 0 | - | 0 | 4 | - | 4 | 4 | - |
| Ramka 2 |  | 2 | 1 | - | - | 1 | 1 | 1 | - | 1 | 1 | - | 2 | 2 | 2 | - | 5 | 5 | - | 6 | 0 | - |

ilość błędów: ilość nie pustych kolumn, 15

**4.4)** W trakcie wykonywania pewnego 700-bajtowego programu zaobserwowano następujący ciąg odwołań do pamięci (podane liczby wskazują adresy komórek pamięci - nie są to numery stron):

251, 46, 231, 150, 250, 120, 140, 32, 85, 231, 220, 173, 41, 309, 151, 245, 320, 176, 234, 256, 234, 164

Wyznacz ciąg odwołań do stron, zakładając, że rozmiar strony wynosi **100 bajtów**, Następnie, zakładając, że programowi temu przydzielono w pamięci **200 bajtów**, wyznacz liczby błędów braku stron dla algorytmu **LRU**. Należy podać sposób rozwiązania i uzasadnienie. Przedstawić zmiany zawartości ramek przydzielonych procesowi.

ilość ramek = 200 bajtów / 100 bajtów = 2

każda strona = 100 bajtów, cały program ma 700, także

strona 0: 0-99 bajtów

strona 1: 100-199 bajtów

strona 2: 200-299 bajtów

strona 3: 300-399 bajtów

strona 4: 400-499 bajtów

strona 5: 500-599 bajtów

strona 6: 600-699 bajtów

przykład przeliczenia adresu na stronę: 410 jest w zasięgu 400-499, zatem 4 strona etc.

410, 245, 173, 450, 120, 231, 610, 309, 176, 234, 430, 185, 210, 551, 31, 245, 556, 434, 565, 650, 46, 458

| komórki | 410 | 245 | 173 | 450 | 120 | 231 | 610 | 309 | 176 | 234 | 430 | 185 | 210 | 551 | 31 | 245 | 556 | 434 | 565 | 650 | 46 | 458 |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| strony: | 4 | 2 | 1 | 4 | 1 | 2 | 6 | 3 | 1 | 2 | 4 | 1 | 2 | 5 | 0 | 2 | 5 | 4 | 5 | 6 | 0 | 4 |

numery stron: 4 2 1 4 1 2 6 3 1 2 4 1 2 5 0 2 5 4 5 6 0 4

Optymalny (zastąp stronę, która najdłużej nie będzie używana)

|  | 4 | 2 | 1 | 4 | 1 | 2 | 6 | 3 | 1 | 2 | 4 | 1 | 2 | 5 | 0 | 2 | 5 | 4 | 5 | 6 | 0 | 4 |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Ramka 1 | 4 | 4 | 4 | - | - | 2 | 6 | 3 | - | 2 | 4 | - | 4 | 5 | 0 | - | 0 | 4 | - | 4 | 4 | - |
| Ramka 2 |  | 2 | 1 | - | - | 1 | 1 | 1 | - | 1 | 1 | - | 2 | 2 | 2 | - | 5 | 5 | - | 6 | 0 | - |

ilość błędów: ilość nie pustych kolumn, 15