1. Model programowania SPMD (single program multiple data):
a) może być realizowany tylko na maszynach SIMD
b) może być realizowany i w MPI, i w OpenMP
c) zawsze polega na wzbogaceniu kodu sekwencyjnego dyrektywami kompilatora
d) w standardowych środowiskach programowania jest ogólniejszy niż MPMD (każdy program MPMD można sprowadzić do SPMD)
e) wymaga, aby każda linijka kodu była wykonywana przez wszystkie procesy (wątki), tyle że pracujące na różnych danych

2. Wypisz zależności pojawiające się w przykładowym kodzie:

```
y += 4 * sin (i * 3.14);
A [i] = 3 * x + z * y;
z = x * y
```

nr linii	nr linii	zmienna	typ zależności
1	2	Y	RAW
1	3	Y	RAW
2	3	Z	WAR

3. Przeanalizuj zależności w przykładowym kodzie. Jeżeli w kodzie występuje określona zależność przenoszona w pętli wpisz obok typu zależności nazwę tablicy, której dotyczy; jeśli danej zależności nie ma w kodzie, wpisz kreskę:

```
for (i = 1; i < N; ++i)
{
        A [i] = D [i + 1] – C [i – 1];
        D [i + 1] = 2 * D [i – 1];
    }
```

zależności wyjścia (WAW): -
anty-zależności (WAR): -
zależności rzeczywiste (RAW): D

4. Operacją komunikacji grupowej, która przekształca stan pamięci procesów:
P1 (rank = 0): a = {11, 12, 13},b = {0, 0, 0}
P2 (rank = 1): a = {21, 22, 23},b = {0, 0, 0}
P3 (rank = 2): a = {31, 32, 33},b = {0, 0, 0}

w stan:
P1 (rank = 0): a = {11, 12, 13},b = {21, 0, 0}
P2 (rank = 1): a = {21, 22, 23},b = {22, 0, 0}
P3 (rank = 2): a = {31, 32, 33},b = {23, 0, 0}

Jest:
a. MPI_Gather (a, 1, MPI_INT, b, 1, MPI_INT, 1, MPI_COMM_WORLD);
b. MPI_Allgather (a, 1, MPI_INT, b, 1, MPI_INT, MPI_COMM_WORLD);
c. MPI_Scatter (a, 1, MPI_INT, b, 1, MPI_INT, 1, MPI_COMM_WORLD);
d. MPI_Alltoall (a, 1, MPI_INT, b, 1, MPI_INT, MPI_COMM_WORLD);
e. MPI_Reduce (a, b, 1, MPI_INT, MPI_MAX, 1, MPI_COMM_WORLD);
f. MPI_Allreduce (a, b, 1, MPI_INT, MPI_MAX, MPI_COMM_WORLD);
g. MPI_Allreduce (a, b, 1, MPI_INT, MPI_MIN, MPI_COMM_WORLD);

5. Stosując notację taką jak w poprzednim zadaniu i zakładając, że stan pamięci przed wykonaniem operacji był następujący:

P1 (rank = 0): a = {11, 12, 13},b = {0, 0, 0}
P2 (rank = 1): a = {21, 22, 23},b = {0, 0, 0}
P3 (rank = 2): a = {31, 32, 33},b = {0, 0, 0}
zilustruj stan pamięci procesów po wykonaniu operacji:
MPI_REDUCE (a, b, 2, MPI_INT, MPI_MIN, 1, MPI_COMM_WORLD);

P1 (rank = 0): a = {11, 12, 13}, b = {0, 0, 0}
P2 (rank = 1): a = {21, 22, 23}, b = {11, 12, 0}
P3 (rank = 2): a = {31, 32, 33}, b = {0, 0, 0}

6. Standardowe programy w stosunku do programu rozważanego w analizie Amdahla mają najczęściej:
a. mniejszy udział części nie dającej się zrównoleglić
b. większy udział części nie dającej się zrównoleglić
c. mniejszy czas komunikacji
d. większy czas komunikacji
e. gorsze przyspieszenie obliczeń równoległych (dla dużych p)
f. lepsze przyspieszenie obliczeń równoległych (dla dużych p)

7. Sposób podziału iteracji równoległej pętli for pomiędzy wątki przy zastosowaniu klauzuli schedule (dynamic) oznacza, że:
a. przydział będzie dokonywany w trakcie działania programu
b. jeden wątek może dostać wszystkie iteracje
c. liczba iteracji przydzielanych poszczególnym wątkom może być różna
d. narzut związany z wykonaniem równoległym będzie duży
e. rozmiar porcji będzie zmienny (określany dynamicznie)
f. rozmiar porcji będzie równy 1
g. sposób podziału w ostateczności zależeć będzie od wartości odpowiedniej zmiennej środowiskowej

8. Klauzula firstprivate oznacza, że objęta nią zmienna:
a. będzie prywatna dla pierwszego wątku
b. będzie pierwszą zmienną prywatną wątków
c. będzie zmienną prywatną wątków inicjalizowaną jako pierwsza
d. będzie zmienną prywatną wątków, inicjalizowaną wartością sprzed rozpoczęcia wykonywania dyrektywy
e. będzie zmienną prywatną wątków, której wartość z pierwszego wątku zostanie skopiowana do wątku głównego po zakończeniu wykonywania dyrektywy

9. W analizie Gustafsona rozważa się zadania, których czas rozwiązania programem równoległym przy rosnącej liczbie procesorów jest stały, albowiem:
a. rozmiar zadania rośnie wraz z liczbą procesorów (rośnie więc czas rozwiązania na jednym procesorze)
b. czas komunikacji rośnie wolniej niż czas obliczeń
c. udział procentowy części sekwencyjnej w czasie rozwiązania na jednym procesorze maleje wraz ze wzrostem rozmiaru zadania
d. część równoległa osiąga przyspieszenie ponadliniowe

13. Kod:
#pragma omp parallel num_threads (4)
#pragma omp for schedule (static, 3)
for (i = 0l i < 15; ++i) {...}
powoduje rozdzielenie iteracji równoległej pętli for pomiędzy wątki w następujący sposób (napisać jak)
W0: 0, 1, 2, 12, 13, 14
W1: 3, 4, 5
W2: 6, 7, 8
W3: 9, 10, 11

14. Monitor w programowaniu współbieżnym:
a. jest strukturą danych zawierającą informacje o sekcjach krytycznych
b. służy do monitorowania ewentualnych konfliktów wątków w dostępie do pamięci wspólnej
c. jest strukturą danych gwarantującą wzajemne wykluczanie przy realizacji jego procedur
d. jest jednym z mechanizmów wbudowanych w model obiektów Javy
e. jest jednym z mechanizmów wbudowanych w model muteksów pthreads

18. Jakie będą wartości zmiennych a, b i c w dowolnym wątku po wykonaniu następującego fragmentu kodu (wciąż wewnątrz obszaru równoległego) (jeśli nie wiadomo wstaw „?”)?

```
int a = 3 , int b = 1 , int c = 2;
#pragma omp parallel firstprivate (c) num_threads (4)
{
 int b = a + c;
   #pragma omp barrier
   #pragma omp atomic
   c += 1;
   a += 2;
   b += 3;
}
```
a = ?, b = 8, c= 3

19. Operacją komunikacji grupowej, która przekształca stan pamięci procesów:
P1 (rank = 0): a = {11, 12, 13},b = {0, 0, 0}
P2 (rank = 1): a = {21, 22, 23},b = {0, 0, 0}
P3 (rank = 2): a = {31, 32, 33},b = {0, 0, 0}

w stan:
P1 (rank = 0): a = {11, 12, 13},b = {11, 21, 31}
P2 (rank = 1): a = {21, 22, 23},b = {11, 21, 31}
P3 (rank = 2): a = {31, 32, 33},b = {11, 21, 31}

Jest:
a. MPI_Gather (a, 1, MPI_INT, b, 1, MPI_INT, 1, MPI_COMM_WORLD);
b. MPI_Allgather (a, 1, MPI_INT, b, 1, MPI_INT, MPI_COMM_WORLD);
c. MPI_Scatter (a, 1, MPI_INT, b, 1, MPI_INT, 1, MPI_COMM_WORLD);
d. MPI_Alltoall (a, 1, MPI_INT, b, 1, MPI_INT, MPI_COMM_WORLD);
e. MPI_Reduce (a, b, 1, MPI_INT, MPI_MAX, 1, MPI_COMM_WORLD);
f. MPI_Allreduce (a, b, 1, MPI_INT, MPI_MAX, MPI_COMM_WORLD);
g. MPI_Allreduce (a, b, 1, MPI_INT, MPI_MIN, MPI_COMM_WORLD);

20. Stosując notację taką jak w poprzednim zadaniu i zakładając, że stan pamięci przed wykonaniem operacji był następujący:
P1 (rank = 0): a = {11, 12, 13},b = {0, 0, 0}
P2 (rank = 1): a = {21, 22, 23},b = {0, 0, 0}
P3 (rank = 2): a = {31, 32, 33},b = {0, 0, 0}
zilustruj stan pamięci procesów po wykonaniu operacji:
MPI_Reduce (a, b, 1, MPI_INT, MPI_MAX, 1, MPI_COMM_WORLD);

P1 (rank = 0): a = {11, 12, 13}, b = {0, 0, 0}
P2 (rank = 1): a = {21, 22, 23}, b = {31, 0, 0}
P3 (rank = 2): a = {31, 32, 33}, b = {0, 0, 0}

21. Kod:
#pragma omp parallel num_threads (3)
#pragma omp for schedule (static, 4)
for (i = 0l i < 15; ++i) {...}
powoduje rozdzielenie iteracji równoległej pętli for pomiędzy wątki w następujący sposób (wypisać):

W0: 0,1,2,3, 12,13,14
W1: 4,5,6,7
W2: 8, 9, 10, 11

22. Jakie będą wartości zmiennych a, b i c w dowolnym wątku po wykonaniu następującego fragmentu kodu (wciąż wewnątrz obszaru równoległego) (jeśli nie wiadomo wstaw „?”)?

```
int a = 2, int b = 3, int c = 4;
#pragma omp parallel firstprivate (a) num_threads (4)
{
 int b = a + c;
   #pragma omp barrier
   #pragma omp atomic
   c += 1;
   a += 2;
   b += 3;
}
```

a = 4, b = 9, c = ?

23. W wyniku wykonania procedury systemowej fork powstają dwa procesy realizujące ten sam kod, które:
a) posiadają wspólne zmienne globalne
b) posiadają wspólne zmienne lokalne
c) otrzymują jako wartość zwracaną funkcji fork nazwajem swoje identyfikatory
d) są w pełnie niezależne i nie mogą być synchronizowane
e) pełnią różne role: jeden jest procesem nadrzędnym, drugi potomnym
f) proces nadrzędny może synchronizować swoje działanie z działaniem procesu potomnego

24. Przetwarzanie w przeplocie oznacza sytuację kiedy:
a) system operacyjny przydziela wątki tego samego zadania różnym rdzeniom (procesom)
b) system operacyjny realizuje przetwarzanie współbieżne na jednym procesorze (rdzeniu)
c) procesor(rdzeń) stosuje tzw. simultaneous multithreading
d) pojedynczy procesor (rdzeń) na przemian wykonuje fragmenty wielu wątków
e) pojedynczy proces korzysta na przemian z wielu rdzeni

25. Argumentami procedury tworzenia nowego wątku pthread_create są m.in.:
a) nazwa funkcji, którą zacznie wykonywać wątek – funkcji startowej wątku (będąca w rzeczywistości wskaźnikiem do funkcji)
b) rozmiar stosu przydzielonego wątkowi
c) zadany przez użytkownika identyfikator wątku
d) obiekt określający atrybuty (m. in. sposób funkcjonowania) wątku
e) argument dla funkcji startowej wątku będący wskaźnikiem

26. Cechami charakterystycznymi funkcjonowania wątków Pthreads są m.in.
a) istnienie odrębnych funkcji tworzenia i uruchamiania wątków
b) brak możliwości zmiany domyślnego sposobu funkcjonowania wątków w trakcie ich tworzenia
c) konieczność używania jako funkcji startowej wątków, funkcji o tylko jednym argumencie
d) posługiwanie się identyfikatorami wątków tożsamymi z identyfikatorami z systemu operacyjnego

27. Każdy wątek Pthreads posiada własne, niezależne od innych wątków:
a) Każdy proces ma swoją przestrzeń adresową;
b) Każdy wątek ma własny zestaw rejestrów i własny stos;
d) ??
e) ??
f) ??

28. Przyspieszenie równoległe programu rozwiązującego pewien problem jako funkcję liczby procesorów p można zdefiniować jako stosunek:
a) czasu rozwiązania problemu najlepszym programem sekwencyjnym do czasu rozwiązania problemu p-razy większego rozważanym programem na p procesorach
b) czasu pracy jednego procesora przy rozwiązaniu problemu rozważanym programem na p procesorach do czasu pracy na wszystkich procesorach
c) czasu rozwiązania problemu najlepszym programem sekwencyjnym do czasu rozwiązania problemu rozważanym programem na p procesorach

29. Liniowe (idealne) przyspieszenie obliczeń równoległych można scharakteryzować jako:
a) nie dającego się nigdy przewyższyć
b) przyspieszenie w sytuacji gdy czas obliczeń równoległych na p procesorach jest p razy krótszy niż czas obliczeń sekwencyjnych
c) przewidywane przez analizę Amdhala

30. Przyspieszenie programu rozważanego w analize Amdhala ulega nasyceniu (przestaje rosnąć) mimo zwiększajacej się liczby procesów ponieważ program:
a) posiada część sekwencyjną, którą zawsze musi wykonać tylko jeden proces (wątek)
b) ma rozmiar rosnący wraz z liczbą procesorów
c) wykazuje duży narzut na komunikację

31. Błedne założenie (w stosunku do praktyki stosowania rzeczywistych programów równoległych) w ramach analizy Amdhala polega na rozważaniu:
a) zbyt dużej liczbie procesorów
b) zadań zbyt trudnych do zrównoleglenia
c) zadań o stałym rozmiarze przy rosnącej liczbie procesorów

49. Współbieżność wykonania programów P1 i P2 oznacza:
a) wykonanie równoległe (wymaga systemu wieloprocesorowego)
b) nakładanie się czasów wykonania (możliwe wykonanie w przeplocie na jednym procesorze)
c) posiadanie wspólnej przestrzeni adresowej przez P1 i P2
d) konieczność zarządzania przez system operacyjny dostępem do urządzeń wejścia/wyjścia przez P1 i P2
e) żadne z powyższych stwierdzeń nie jest prawdziwe

50. Współbieżność wykonania programów wprowadzona została w celu:

a) umożliwienia działania procesorów wielordzeniowych
b) usprawnienia pracy komputera przy realizacji operacji wejścia/wyjścia
c) szybszej obsługi połączeń sieciowych
d) umożliwienia funkcjonowania procesów wielowątkowych
e) żadne z powyższych stwierdzeń nie jest prawdziwe

51. Proces od wątku różni się m.in.:
a) posiadaniem własnego zestawu rejestrów w trakcie wykonania (wątek współdzieli rejestry z innymi wątkami tego samego procesu)
b) posiadaniem własnej przestrzeni adresowej (wątek współdzieli przestrzeń adresową z innymi wątkami tego samego procesu)
c) posiadaniem bardziej rozbudowanej struktury umożliwiającej zarządzanie przez system operacyjny
d) posiadaniem wyższego priorytetu wykonania
e) żadne z powyższych stwierdzeń nie jest prawdziwe

52. W skład narzędzi programowania OpenMP wchodzą:
a) dyrektywy kompilatora
b) funkcje biblioteczne
c) predefiniowane obiekty (struktury)
d) typy danych
e) zmienne środowiskowe
f) żadna odpowiedź nie jest prawidłowaamdh

53. W trakcie wykonania programu OpenMP obszar równoległy:
a) zaczyna się po dyrektywie parallel
b) zaczyna się po dowolnej dyrektywie podziału pracy
c) oznacza, że program może być wykonywany wielowątkowo (w przeciwieństwie do obszaru sekwencyjnego)
d) oznacza, że zmienne prywatne funkcjonują w wielu kopiach (po jednej dla każdego wątku)
e) żadna odpowiedź nie jest prawidłowa

54. Liczba wątków tworzonych przy wchodzeniu do obszaru równoległego może (jeśli system pozwala) zostać jawnie określona za pomocą:
a) klauzuli num_threads dyrektywy parallel
b) klauzuli num_threads dowolnej dyrektywy podziału pracy
c) procedury omp_set_num_threads wewnątrz obszaru równoległego
d) procedury omp_set_num_threads przed wejściem do obszaru równoległego
e) zmiennej środowiskowej OMP_NUM_THREADS ustawianej przed uruchomieniem programu
f) zmiennej środowiskowej OMP_NUM_THREADS ustawianej przed wejściem przez program do obszaru równoległego
g) żadna odpowiedź nie jest prawidłowa

55. Zmiana wartości zmiennej wspólnej w OpenMP:
a) musi odbywać się w sekcji krytycznej
b) może, a czasami powinna odbywać się w sekcji krytycznej
c) może być dokonana przez dowolny wątek w obszarze równoległym
d) może zostać zrealizowana niepodzielnie (dyrektywa atomic)
e) dokonana przez jeden wątek jest widoczna dla wszystkich wątków
f) żadna odpowiedź nie jest prawidłowa

56. Zmiana wartości zmiennej prywatnej w OpenMP:
a) musi odbywać się w sekcji krytycznej
b) powinna odbywać się w sekcji krytycznej
c) może być dokonana przez dowolny wątek w obszarze równoległym
d) dokonana przez jeden wątek jest widoczna dla wszystkich wątków
e) może zostać zrealizowana niepodzielnie (dyrektywa atomic)
f) żadna odpowiedź nie jest prawidłowa

57. Zmienna lokalna funkcji staje się zmienną prywatną wątków w obszarze równoległym jeśli w kodzie:
a) jest objęta dyrektywą threadprivate w dowolnej funkcji
b) jest objęta dyrektywą threadprivate w tej samej funkcji, w której znajduje się dyrektywa parallel
c) jest objęta jedną z klauzul private, firstprivate itp. w dyrektywie parallel danego obszaru równoległego
d) jest objęta jedną z klauzul private, firstprivate itp. w dyrektywie parallel danego obszaru równoległego w danej funkcji
e) jest zmienną sterującą pętli for
f) jest zmienną sterującą równoległej pętli for
g) żadna odpowiedź nie jest prawidłowa

58. Zakończenie operacji wysyłania danych procedurą MPI_Send (powrót z procedury) lub MPI_Isend (np. powrót z procedury MPI_Wait) oznacza zawsze, że:
a) dane dotarły do adresata
b) system odnalazł adresata gotowego do odebrania komunikatu (adresata, który wywołał procedurę MPI_Recv z pasującymi argumentami)
c) system skopiował dane do wewnętrznego bufora przesyłania danych
d) obszary danych objęte poleceniem wysyłania mogą być zmieniane, co nie spowoduje zmiany zawartości komunikatu
e) system potwierdził rozpoczęcie operacji przesyłania danych
f) żadna z powyższych odpowiedzi nie jest prawidłowa

59. Po powrocie z procedury odbierania nieblokującego MPI_Irecv(&a, …… &req), gdzie a oznacza pewną zmienną, mamy pewność, że wartość a jest wartością otrzymaną w komunikacie P:
a) od razu
b) dopiero po sprawdzeniu zawartości obiektu req (type MPI_Request)
c) dopiero po powrocie z procedury MPI_Wait (& req, & stat)
d) dopiero po powrocie z procedury MPI_Wait (& req, & stat) i sprawdzeniu odpowiedniej zmiany wartości obiektu stat
e) dopiero po powrocie z procedury MPI_Test (& req, & flag, & stat)
f) dopiero po powrocie z procedury MPI_Test (& req, & flag, & stat) i sprawdzeniu odpowiedniej zmiany wartości obiektu stat
g) dopiero po powrocie z procedury MPI_Test (& req, & flag, & stat) i sprawdzeniu odpowiedniej zmiany wartości zmiennej flag
h) żadna z powyższych odpowiedzi nie jest prawidłowa

60. Komunikator w MPI:
a) jest konieczny tylko do przeprowadzenia komunikacji grupowej
b) jest konieczny do realizacji dowolnego przesyłania komunikatów
c) oznacza grupę procesów i związane z nią informacje umożliwiające wymianę komunikatów
d) oznacza proces pośredniczący w wymianie komunikatów
e) jest reprezentowany zawsze przez obiekt MPI_COMM_WORLD
f) jest zawsze tylko jeden w programie
g) jest co najmniej jeden w programie; jeśli jeden, to jest to MPI_COMM_WORLD
h) jest co najmniej jeden w programie, zawsze typu MPI_Comm
i) żadne z powyższych twierdzeń nie jest prawdziwe

61. Zakładając początkowy stan zmiennych w kolejnych procesach w postaci:
P1 (rank = 0), a = {11, 12, 13}, b = {0, 0, 0}
P2 (rank = 1), a = {21, 22, 23}, b = {0, 0, 0}
P3 (rank = 2), a = {31, 32, 33}, b = {0, 0, 0}
operacja
MPI_Scatter (a, 1, MPI_INT, b, 1, MPI_INT, 2, MPI_COMM_WORLD);
prowadzi do stanu zmiennej b:
a) P1: {11, 12, 13}, P2: {21, 22, 23}, P3: {31, 32, 33}
b) P1: {11, 12, 13}, P2: {11, 12, 13}, P3: {11, 12, 13}
c) P1: {31, 32, 33}, P2: {31, 32, 33}, P3: {31, 32, 33}
d) P1: {11, 0, 0}, P2: {12, 0, 0}, P3: {13, 0, 0}
e) P1: {11, 12, 13}, P2: {12, 0, 0}, P3: {13, 0, 0}
f) P1: {31, 0, 0}, P2: {32, 0, 0}, P3: {33, 0, 0}
g) żadna odpowiedź nie jest prawidłowa

62. Operacją komunikacji grupowej, która przekształca stan pamięci procesów:
P1 (rank = 0): a = {11, 12, 13}, b = {0, 0, 0}
P2 (rank = 1): a = {21, 22, 23}, b = {0, 0, 0}
P3 (rank = 2): a = {31, 32, 33}, b = {0, 0, 0}
w stan
P1 (rank = 0): a = {11, 12, 13}, b = {31, 0, 0}
P2 (rank = 1): a = {21, 22, 23}, b = {31, 0, 0}
P3 (rank = 2): a = {31, 32, 33}, b = {31, 0, 0}
jest:
a) MPI_Gather (a, 1, MPI_INT, b, 1, MPI_INT, 2, MPI_COMM_WORLD);
b) MPI_Allgather (a, 1, MPI_INT, b, 1, MPI_INT, 2, MPI_COMM_WORLD);
c) MPI_Scatter (a, 1, MPI_INT, b, 1, MPI_INT, 2, MPI_COMM_WORLD);
d) MPI_Alltoall (a, 1, MPI_INT, b, 1, MPI_INT, MPI_COMM_WORLD);
e) MPI_Reduce (a, b, 1, MPI_INT, MPI_SUM, 2, MPI_COMM_WORLD);
f) MPI_Allreduce (a, b, 1, MPI_INT, MPI_SUM, MPICOMM_WORLD);
g) MPI_Allreduce (a, b, 1, MPI_INT, MPI_MAX, MPI_COMM_WORLD); (!)

63. Jakie będą wartości zmiennych a, b i c po wykonaniu następującego fragmentu kodu:

```
int a = 1; int b = 2; int c = 4;
#pragma omp threadprivate b
#pragma omp parallel firstprivate(c) num_threads(4)
{
   int d = a + c;
   #pragma omp atomic
   c += 1;
   #pragma omp atomic
   a += 2;
   #pragma omp atomic
   d += 3;
   b = d;
}
```

a) a = 3, b = 8, c = 4
b) a = 3, b = 8, c = 5
c) a = 3, b = 8, c = 5
d) a = 3, b = 8, c = 8
e) a = 9, b = 8, c = 4
f) a = 9, b = 8, c = 5
g) a = 9, b = 8, c = 8
h) a = 3, b = 17, c = 4
i) a = 3, b = 17, c = 5
j) a = 3, b = 17, c = 8
k. a = 9, b = 17, c = 4
l. a = 9, b = 17, c = 5
m. a = 9, b = 17, c = 8
n. żadna odpowiedź nie jest prawidłowa

64. Jakie wartości zmiennych a, b i c zostaną wypisane przez polecenie printf po wykonaniu następującego fragmentu kodu:

```
int a = 1; int b = 2; int c = 3;
#pragma omp threadprivate(b)
#pragma omp parallel firstprivate (a) num_threads (4)
{
   int d = a + c;
   #pragma omp atomic
   c += 1;
   #pragma omp atomic
   a += 2;
   #pragma omp atomic
   d += 3;
   b = d;
   printf ("a = %d, b = %d, c = %d\n", a, b, c);
}
```

a) a = 9, b = 16, c = 4
b) a = 9, b = 16, c = 7
c) a = 9, b = 7, c = 4
d) a = 9, b = 7, c = 7
e) a = 9, b = 16, c = 4
f) żadna odpowiedź nie jest prawdłowa

65. Jakie wartości zmiennych a, b i c zostaną wypisane przez polecenie printf po wykonaniu następującego fragmentu kodu:

```
int a = 1; int b = 2; int c = 3;
#pragma omp threadprivate (b)
#pragma omp parallel firstprivate (c) num_threads (4)
{
   int d = a + c;
   #pragma omp atomic
   c += 1;
   #pragma omp atomic
   a += 2;
   #pragma omp atomic
   d += 3;
   b =d;
   printf ("a = %d, b = %d, c = %d\n", a, b, c);
}
```

a) a = 3, b = 16, c = 7;
b) a = 3, b = 16, c = 4;
c) a = 9, b = 7, c = 7;
d) a = 9, b = 7, c = 4;
e) a = 9, b = 7, c = 7;
f) żadna odpowiedź nie jest prawidłowa

66. Jakie będą wartości zmiennych a, b i c po wykonaniu następującego fragmentu kodu:

```
int a = 1, int b = 2, int c = 3;
#pragma omp threadprivate (b)
#pragma omp parallel firstprivate (c) num_threads (4)
{
   int d = a + c;
   #pragma omp atomic
   c += 1;
   #pragma omp atomic
   a += 2;
   #pragma omp atomic
   d += 3;
   b = d;
}
```

a) a = 3, b = 7, c = 3
b) a = 3, b = 7, c = 4
c) a = 3, b = 7, c = 7
d) a = 9, b = 7, c = 3
e) a = 9, b = 7, c = 4
f) a = 9, b = 7, c = 7
g) a = 3, b = 16, c = 3
h) a = 3, b = 16, c = 4
i) a = 3, b = 16, c = 7
j) a = 9, b = 16, c = 3
k. a = 9, b = 16, c = 4
l. a = 9, b = 16, c = 7
m. żadna odpowiedź nie jest prawidłowa

67. Kod:
#pragma omp parallel for schedule (static, 3)
for (i = 0; i < 13; ++i) {...}
powoduje rozdzielenie iteracji równoległej pętli for pomiędzy 3 wątki (W1, W2, W3) wykonujące program w następujący sposób:
a) W1 – 0, 3, 6, 9, 12; W2 – 1, 4, 7, 10; W3 – 2, 5, 8, 11
b) W1 – 0, 1, 6, 7, 12; W2 – 2, 3, 8, 9; W3 – 4, 5, 10, 11
c) W1 – 0, 1, 2, 9, 10, 11; W2 – 3, 4, 5, 12; W3 – 6, 7, 8
d) W1 – 0, 1, 2, 9, 10; W2 – 3, 4, 5, 11; W3 – 5, 7, 8, 12
e) W1 – 0, 1, 2, 3, 4; W2 – 5, 6, 7, 8, 9; W3 – 10, 11, 12

68. Kod:
#pragma omp parallel for schedule (static, 2)
for (i = 0; i < 13; ++i) {...}
powoduje rozdzielenie iteracji równoległej pętli for pomiędzy 3 wątki (W1, W2, W3) wykonujące program w następujący sposób:
a) W1 – 0, 3, 6, 9, 12; W2 – 1, 4, 7, 10; W3 – 2, 5, 8, 11;
b) W1 – 0, 1, 6, 7, 12; W2 – 2, 3, 8, 9; W3 – 4, 5, 10, 11;
c) W1 – 0, 1, 2, 9, 10, 11; W2 – 3, 4, 5, 12; W3 – 6, 7, 8;
d) W1 – 0, 1, 2, 9, 10; W2 – 3, 4, 5, 11; W3 – 6, 7, 8, 12;
e) W1 – 0, 1, 2, 3, 4; W2 – 5, 6, 7, 8, 9; W3 – 10, 11, 12;

69. Jakie zależności występują w poniższym kodzie (pominięto inicjalizowanie zmiennych):
```
for (i = 2; i < N; ++i)
{
   A [i] = B [i + 1] - C [i - 2];
   A [i + 2] = 2 * D [i - 1];
}
```
a) występują zależności wyjścia (WAW)
b) występują anty-zależności (WAR)
c) występują zależności rzeczywiste (RAW)
d) występują zaleności wyjścia(WAW) przenoszone w pętli
e) występują anty-zależności (WAR) przenoszone w pętli
f) występują zależności rzeczywiste (RAW) przenoszone w pętli

70. W przykładowym kodzie (pominięte inicjowanie zmiennych):
```
for (i = 0; i < N; ++i)
{
   A [i]  =B [i + 1] – C [i - 2];
   D [i + 2] = 2 * A [i - 1];
}
```
a) występują zależności wyjścia (WAW)
b) występują anty-zależności (WAR)
c) występują zależności rzeczywiste (RAW)
d) występują zależności wyjścia (WAW) przenoszone w pętli
e) występują anty-zależności (WAR) przenoszone w pętli
f) występują zależności rzeczywiste (RAW) przenoszone w pętli
g) nie ma zależności

71. W przykładowym kodzie (pominięto inicjalizowanie zmiennych)

```
for (i = 0; i < N; ++i)
{
   x += 2 * y + z;
   A [i] = 3 * x + w * y;
   y = 4 * i;
}
```

a) występują zależności wyjścia (WAW)
b) występują anty-zależności (WAR)
c) występują zależności rzeczywiste (RAW)
d) występują zależności wyjścia (WAW) przenoszone w pętli
e) występują anty-zależności (WAR) przenoszone w pętli
f) występują zależności rzeczywiste (RAW) przenoszone w pętli
g) nie ma zależności

72. Umieszczenie jakiej pętli w połączeniu ze zmianą jednej z linijek kodu wewnątrz pętli usuwa zależność przenoszoną w następującej pętli (pominięte szczegóły na początku i końcu pętli):

```
for (i = 0; i < N; ++i)
{
 A [i] = B [i + 1] - C [i - 1];
 D [i] = A [i];
}
```

a) for (i = 0; i < N; ++i) E [i] = A [i]; oraz zmiana E [i] = B [i + 1] - C [i - 1];
b) for (i = 0; i < N; ++i) E [i] = A [i]; oraz zmiana D [i + 1] = E [i + 1]
c) for (i = 0; i < N; ++i) E [i] = A [i]; *+oraz zmiana D [i + 1] = E [i]
d) for (i = 0; i < N; ++i) E [i] = A [i]; oraz zmiana D [i + 1] = E [i - 1]
e) for (i = 0; i < N; ++i) E [i] = A [i]; oraz zmiana D [i] = E [i]
f) w pętli jest zależność nieusuwalna
g) w pętli nie ma żadnej przenoszonej zależności

73. Umieszczenie jakiej pętli w połączeniu ze zmianą jednej z linijek kodu wewnątrz pętli usuwa zależność przenoszoną w następującej pętli (pominięte szczegóły na początku i na końcu pętli):

```
for (i = 0; i < N; ++i)
{
 A [i] = B [i + 1] – C [i - 1];
 D [i + 1] = 2 * A [i - 1];
}
```

a) for (i = 0; i < N; ++i) E [i] = A [i];
oraz zmiana E [i] = B [i + 1] - C [i - 1];
b) for (i = 0; i < N; ++i) E [i] = A [i];
oraz zmiana D [i + 1] = E [i + 1]
c) for (i = 0; i < N; ++i) E [i] = A [i];
oraz zmiana D [i + 1] = E [i]
d) for (i = 0; i < N; ++i) E [i] = A [i];
oraz zmiana D [i + 1] = E [i - 1]
e) for (i = 0; i < N; ++i) E [i] = A [i];
oraz zmiana D [i] = E [i]
f) w pętli jest zależność nieusuwalna
g) w pętli nie ma żadnej przenoszonej zależności

84. Przyspieszenie programu rozważanego w analizie Amdahla ulega nasyceniu (przestaje rosnąć) mimo zwiększającej się liczby procesorów ponieważ program:
a. posiada część sekwencyjną, którą zawsze musi wykonywać tylko jeden proces (wątek)
b. ma rozmiar rosnący wraz z liczbą procesorów
c. wykazuje duży narzut na komunikację

85. Błędne założenie (w stosunku do praktyki stosowania rzeczywistych programów równoległych) w ramach analizy Amdahla polega na rozważaniu:
a. zbyt dużej liczby procesorów
b. zadań zbyt trudnych do zrównoleglenia
c. zadań o stałym rozmiarze przy rosnącej liczbie procesorów

86. W analizie Gustaffsona rozważa się zadania w których rozmiar rośnie wraz z liczbą procesorów ale czas rozwiązania programem równoległym na p procesorach jest stały albowiem zakłada się że:
a. czas komunikacji rośnie wolniej niż czas obliczeń
b. udział procentowy części sekwencyjnej w czasie rozwiązania na jednym procesorze maleje wraz z wzrostem rozmiaru zadania
c. część równoległa osiąga przyspieszenie ponadliniowe

87. W analizie Amdahla rozważa się zrównoleglenie programu:
a. którego sekwencyjny czas rozwiązania problemu jest najkrótszy z możliwych
b. którego czas wykonania sekwencyjnego jest równy sumie czasu wykonania części sekwencyjnej i części dającej się zrównoleglić
c. który posiada część, którą zawsze musi wykonywać jeden proces (wątek)
d. który posiada część dającą się zrównoleglić z idealną komunikacją
e. który posiada część dającą się zrównoleglić idealnie
f. który daje się wydajniej (efektywniej) zrównoleglić niż zdecydowana większość programów rzeczywistych

88. W analizie Amdahla graniczną wartość przyspieszenia obliczeń , której nie może przekroczy program o czasie wykonywania T(p) = s+r/p jest (w poniższych wzorach f oznacza t.zw. udział części sekwencyjnej f = s/(s+r))
a) f
b) 1+f
c) 1+ 1/f
d) 1/f
e) 1/(1+f)
f) 1/(1+ 1/f)
g) żadna odpowiedź nie jest prawidłowa

89. Złożoność obliczeniowa algorytmu to:
a. stopień skomplikowania algorytmu
b. wymagany stopień skomplikowania komputera konieczny do realizacji algorytmu
c. ilość zasobów komputera wymaganych do realizacji algorytmu
d. ilość zasobów komputera wymaganych do realizacji algorytmu dla najbardziej wymagających (skomplikowanych) danych
e. ilość zasobów komputera wymaganych do realizacji algorytmu jako funkcja rozmiaru danych wejściowych

90. Złożoność pesymistyczna algorytmu oznacza:
a. złożoność dla największych dopuszczalnych danych
b. złożoność dla najmniej korzystnych warunków realizacji zadania
c. złożoność dla najmniej korzystnych przypadków danych wejściowych
d. złożoność dla najmniej korzystnych architektur procesorów

91. Algorytmy sortowania posiadają złożoność czasową:
a. liniową (problem „łatwy”)
b. wielomianową (problem „łatwy”)
c. wykładniczą (problem „trudny”)
d. silnia (problem „trudny”)
e. nie można określić nie podając o jaką złożoność chodzi (optymistyczną, pesymistyczną czy oczekiwaną)
f. problem nie posiada rozwiązania, prowadzi do sprzeczności

92. Złożoność czasowa algorytmu obliczania średniej ciągu liczb jest funkcją liczby liczb w ciągu:
a. liniową (problem „łatwy”)
b. wielomianową (problem „łatwy”)
c. wykładniczą (problem „trudny”)
d. silnia (problem „trudny”)
e. nie można określić nie podając o jaką złożoność chodzi (optymistyczną, pesymistyczną czy oczekiwaną)
f. problem nie posiada rozwiązania, prowadzi do sprzeczności

Zadania nadobowiązkowe

15. Środowisko RPC:
a. korzysta zawsze z protokołu TCP/IP
b. może prowadzić system operacyjny do konfliktów przy ustalaniu numerów portów serwera
c. stosuje transformację argumentów przesyłanych przez sieć do wspólnego formatu
d. umożliwia zdalne wywołanie procedur z wieloma argumentami
e. wymusza określanie argumentów jako strumieni bajtów
f. posługuje się mechanizmem wersji programów

16. Model dostępu do pamięci wspólnej UMA (uniform memory access):
a. jest charakterystyczny dla systemów SMP (symmetric multiprocessing)
b. jest charakterystyczny dla systemów DSM (distributed shared memory)
c. oznacza, że zawartość każdej komórki pamięci jest dostępna dla każdego procesora
d. oznacza, że zawartość każdej komórki jest dostępna dla każdego procesora w takim samym czasie

32. Maszyna SIMD to:
a) komputer wektorowy
b) komputer macierzowy
c) komputer wieloprocesorowy
d) komputer SMP

33. SPM – czym się charakteryzuje:
a) niski koszt
b) skalowalność
c) związek z NUMA
d) związek z UMA
e) związek z ccNUMA

34. Architektura komputerów równoległych, w której występuje wiele strumieni danych i jeden strumień rozkazów jest nazywana w klasyfikacji Flynna architekturą:
a) MISD
b) SISD
c) MIMD
d) SIMD
e) MISD
f) SISD
g) MPMD
h) SPMD
i) żadna odpowiedź nie jest prawidłowa

35. Procesory wielordzeniowe:
a) są nazywane inaczej układami scalonymi wieloprocesorowymi
b) pracują w modelu SIMD (single instruction multiple data)
c) nigdy nie przekroczą liczby rdzeni ok. kilkunastu
d) nie posiadają pamięci podręcznej L2 i L3
e) żadna odpowiedź nie jest prawidłowa

36. Współczesne procesory wielordzeniowe realizują model przetwarzania (na poziomie wielu rdzeni, nie na poziomie pojedynczego rdzenia):
a) MISD
b) MIMD
c) SIMD
d) SISD
e) MPMD
f) SPMD
g) żadna odpowiedź nie jest prawidłowa

37. Klastry:
a) są specjalistycznymi superkomputerami
b) powstają przez połączenie wielu komputerów siecią i wyposażenie ich w specjalne oprogramowanie pozwalające traktować je jak pojedynczy system do uruchamiania programów
c) zakładają najczęściej model programowania bez pamięci wspólnej
d) są najdroższymi komputerami równoległymi
e) skalują się (dają się praktycznie wykorzystywać) dla liczb procesorów (rdzeni) do rzędu kilku dziesięciu
f) skalują się (dają się praktycznie wykorzystywać) dla liczb procesorów (rdzeni) do rzędu kilku tysięcy
g) skalują się (dają się praktycznie wykorzystywać) dla liczb procesorów (rdzeni) do rzędu kilkuset tysięcy

38. Klastry realizują model przetwarzania:
a) MISD
b) SISD
c) MIMD
d) SIMD
e) MPMD
f) eSPMD

39. Rozkazy typu SIMD we współczesnych procesorach oznaczają rozkazy:
a) dotyczące wyłącznie grafiki
b) dotyczące zawartości rejestrów zawierających kilka spakowanych liczb
c) wykonywane równolegle na kilku liczbach
d) wykonywane we współpracy z koprocesorem wektorowym
e) żadna odpowiedź nie jest prawidłowa

40. Potokowe przetwarzanie rozkazów oznacza przetwarzanie:
a) potoku rozkazów – kolejny rozkaz po zakończeniu poprzedniego
b) zbliżone do pracy na taśmie produkcyjnej – pojedyncza jednostka funkcjonalna procesora realizuje tylko część przetwarzania rozkazu
c) dzięki któremu procesor może współbieżnie przetwarzać wiele rozkazów
d) wymagające istnienia złożonych procesorów o wielu jednostkach funkcjonalnych
e) takie jak w kartach graficznych (inaczej przetwarzanie strumieniowe)

41. Nazwa NUMA oznacza systemy wieloprocesorowe (wielordzeniowe):
a) z jednolitym dostępem do pamięci (zawartość każdej komórki pamięci operacyjnej dostarczana procesorowi w takim samym czasief)
b) z niejednolitym dostępem do pamięci(zawartości różnych komórek pamięci operacyjnej mogą być dostarczane procesorowi w różnych czasach)
c) skalujące się (dające się praktycznie wykorzystywać) dla liczb procesorów do rzędu kilkunastu
d) skalujące się (dające się praktycznie wykorzystywać) dla liczb procesorów do rzędu kilkudziesięciu
e) skalujące się (dające się praktycznie wykorzystywać) dla liczb procesorów do rzędu kilkuset

42. Model dostępu do pamięci wspólnej ccNUMA (cache-coherent non-uniform memory access):
a) jest charakterystyczny dla systemów SMP (symmetric multiprocessing)
b) jest charakterystyczny dla systemów DSM (distributed shared memory)
c) oznacza że zawartość każdej komórki pamięci jest dostępna dla każdego procesora
d) oznacza że zawartość każdej komórki jest dostępna dla każdego procesora w takim samym czasie
e) żadna odpowiedź nie jest prawdziwa

43. Architektura DSM (distributed shared memory):
a) jest najczęściej związana z modelem dostępu do pamięci UMA (uniform memory access)
b) jest najczęściej związana z modelem dostępu do pamięci NUMA (non-uniform memory access)
c) jest najczęściej związana z modelem dostępu do pamięci ccNUMA (cache-coherent non-uniform memory access)
d) jest najlepiej skalującą się architekturą maszyn z pamięcią wspólną
e) jest najtańszą architekturą komputerów wieloprocesorowych
f) może być stosowana w systemach o liczbie procesorów przekraczającej kilka tysięcy
g) żadna odpowiedź nie jest prawidłowa

44. Nazwa SMP (symmetric multiprocessing) oznacza systemy:
a) jednoprocesorowe
b) wieloprocesorowe, tzw. symetryczne (każdy procesor widzi system tak samo)
c) jednoprocesorowe z kartami graficznymi używanymi do obliczeń ogólnego przeznaczenia
d) z jednolitym dostępem do pamięci UMA (zawartość każdej komórki pamięci operacyjnej dostarczana procesorowi w takim samym czasie)
e) z niejednolitym dostępem do pamięci NUMA (zawartości różnych komórek pamięci operacyjnej mogą być dostarczane procesorowi w różnych czasach)

45. Komputery SMP realizują model przetwarzania:
a) MIMD
b) SIMD
c) MISD
d) SISD

46. Wskaż prawidłową kolejność etapów przetwarzania potokowego (ID – dekodowanie rozkazu, IE – wykonanie rozkazu, IF – pobranie rozkazu, OF – pobranie argumentów, WB – zapis wyniku):
a) IE ID OF IF WB
b) ID IF WB IE OF
c) IE IF WB ID OF
d) WB IE IF ID OF
e) IF ID OF IE WB

47. Rdzeń procesora wielordzeniowego:
a) jest bardzo zbliżony do dawnych procesorów (jednordzeniowych), ale nie potrafi wykonywać bardziej złożonych rozkazów
b) jest bardzo zbliżony do dawnych procesorów (jednordzeniowych), ale bez pamięci podręcznej L2 i L3
c) wydziela znacznie mniej ciepła niż procesor jednordzeniowy o tej samej częstotliwości pracy
d) może mieć własny dostęp do fragmentu pamięci głównej (architektura NUMA)
e) żadne stwierdzenie nie jest prawdziwe

48. Na stosie w trakcie realizacji programu przechowywane są:
a) dane programu
b) dane wspólne procedur
c) dane wejściowe (argumenty) procedur
d) dane prywatne (lokalne) procedur
e) dane do komunikacji procedur z systemem operacyjnym
f) dane do komunikacji międzyprocesowej

74. Szerokość połowienia danego układu (topologii) procesorów oznacza:
a) minimalną liczbę procesorów na przekroju połowiczym układu
b) maksymalną liczbę krawędzi łączących połowę liczby procesorów
c) minimalną liczbę krawędzi, których usunięcie dzieli układ na pół
d) każda odpowiedź jest prawidłowa
e) żadna odpowiedź nie jest prawidłowa

75. Szerokość połowienia danego układu (topologii) procesorów w największym stopniu informuje o:
a) odporności układu na awarie
b) przepustowości układu
c) zdolności efektywnej realizacji rozgłaszania
d) koszcie układu
e) żadna odpowiedź nie jest prawidłowa

76. Liczba krawędzi układu (topologii) procesorów w największym stopniu informuje o:
a) przepustowości układu
b) koszcie układu
c) odporności układu na awarie
d) koszcie stworzenia sieci dla danego układu

77. Średnica topologii hiperkostki dla p procesorów wynosi:
a) log p
b) 1/2 p
c) p
d) p log p
e) 1/2 p log p
f) żadna odpowiedź nie jest poprawna

78. Liczba krawędzi topologii hiperkostki dla p procesów wynosi
a) 2p
b) p*p log p
c) 1/2 log p

79. Średnica topologii torusa 2D dla p procesów wynosi:
a) sqrt(p)
b) 1/2 sqrt(p)
c) p * sqrt(p)
d 1/2 *p * sqrt(p)
e) żadna odpowiedź nie jest prawidłowa

80. Szerokość połowienia topologii torusa 2D dla p procesów wynosi:
a) p
b) 1/2p*sqrt(p)
c) 2*sqrt(p)

81. Połączalność krawędziowa (arc connectivity) danego układu (topologii) procesorów oznacza:
a. maksymalną liczbę krawędzi łączących połowę liczby procesorów
b. minimalną liczbę krawędzi, których usunięcie dzieli układ na pół
c. minimalną liczbę krawędzi, których usunięcie dzieli układ na dwie sieci rozłączne

Historia zmian

v.1.02

- zmiany w zadaniach 1, 2, 9, 14, 27, 32, 48, 49, 51, 52, 57, 58, 60, 90

v.1.01

- zadania nieobowiązkowe przesunięte na koniec
- zmiany w zadaniach: 1, 2, 3, 4, 5, 6, 7, 17, 18, 19, 20, 21, 24, 25, 26, 55, 56, 69, 71, 87
- zlikwidowane powtórzenia zadań: (1, 17), (28, 82), (29, 83)