Kolosy PRiR

1. Makefile albo jakies przydziały

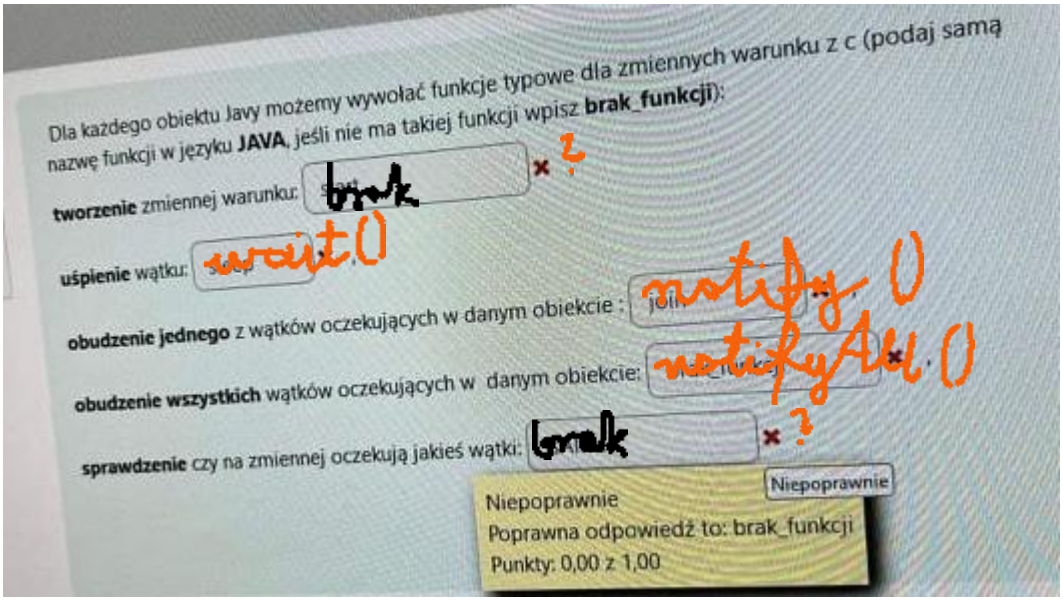
Nie wiem czy ten Makefile jest dobrze

```
# kompilator c
CCOMP = gcc

# konsolidator
LOADER = gcc

# opcje optymalizacji:
# wersja do debugowania
#OPT = -g -DDEBUG
# wersja zoptymalizowana do mierzenia czasu
OPT = -O3

# pliki naglowkowe
#INC = -I../pomiar_czasu

# biblioteki
#LIB = -L../pomiar_czasu -lm

# zaleznosci i komendy
moj_program: moj_program.o pomiar_czasu.o
	$(LOADER) $(OPT) moj_program.o pomiar_czasu.o -o moj_program $(LIB)

# jak uzyskac plik moj_program.o ?
moj_program.o:
	$(CCOMP) -c $(OPT) moj_program.c

pomiar_czasu.o: pomiar_czasu.c
	$(CCOMP) -c $(OPT) pomiar_czasu.c

clean:
	rm -f *.o
~
```

2. Java


Proszę dopasować definicje do metod klasy Thread w JAVA :
void interrupt()
static void sleep(long millis)
void start()
void join()
void run()
Twoja odpowiedź jest częściowo poprawna.
Poprawnie wybrałeś 3.
Poprawna odpowiedź to:
void interrupt() → ustawienie sygnalizatora przerwania wątku,
static void sleep(long millis) → udpienie bieżącego wątku na określony czas,
void start() → polecenie dla JVM rozpoczęcia pracy wątku i wykonania metody run,
void join() → oczekiwanie na zakończenie wątku,
void run() → natychmiastowy powrót, jeśli wątek nie otrzymał w konstruktorze obiektu typu Runnable

3.


Dla każdego obiektu Javy możemy wywołać funkcje typowe dla zmiennych warunku z c (podaj samą nazwę funkcji w języku JAVA, jeśli nie ma takiej funkcji wpisz brak_funkcji):
tworzenie zmiennej warunku:
uśpienie wątku:
obudzenie jednego z wątków oczekujących w danym obiekcie :
obudzenie wszystkich wątków oczekujących w danym obiekcie:
sprawdzenie czy na zmiennej oczekują jakieś wątki:
Niepoprawnie
Poprawna odpowiedź to: brak_funkcji
Punkty: 0,00 z 1,00

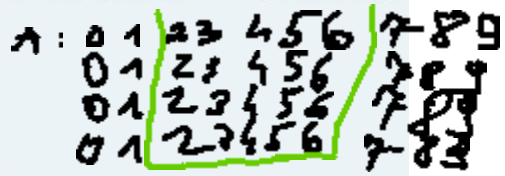
4.

Napisz jakie wartości zawiera tablica B dla procesów o **rank** równym: 0, 1, 2, 3 gdzie liczba procesów **size** = 4, po wykonaniu polecenia **MPI_Allreduce** dla następujących warunków początkowych:

int A[10] = { 0, 0, 0, 0, 0, 0,0, 0, 0, 0 }; for(i=0; i<10; i++) { A[i] = i; }

int B[10] = { 0, 0 ,0, 0, 0, 0, 0, 0, 0, 0 }; for(i=0; i<10; i++) { B[i] = rank; }

MPI_Allreduce(&A[2], &B[1], 5, MPI_INT, MPI_SUM, MPI_COMM_WORLD);



// proces 0: B[0] = 0 B[1] = 8

// proces 1: B[2] = 12 B[3] = 16

// proces 2: B[4] = 20 B[5] = 24

// proces 3: B[6] = 3 B[7] = 3

5.

Zakładając następujące warunki początkowe:

1. Liczba procesów w ramach komunikatora **MPI_COMM_WORLD**: **size = 3**
2. Identyfikator każdego procesu z zakresu od **0** do **size-1** przechowywany w zmiennej: **rank**
3. Tablice **A** i **B** zdefiniowane i zainicjowane kodem:

int A[100] = { 0 }; for(i=0; i<100; i++) { A[i] = 100+10*rank+i; }
int B[100] = { 0 }; for(i=0; i<100; i++) { B[i] = 200+10*rank+i; }

proszę uzupełnić poniższe stwierdzenia dotyczące stanu tablic **A** i **B** po wykonaniu polecenia MPI:

MPI_Alltoall(&B[1], 3, MPI_INT, &A[1], 3, MPI_INT, MPI_COMM_WORLD);

(najlepiej rozwiązać zadanie na kartce i następnie wpisać poprawne wartości)

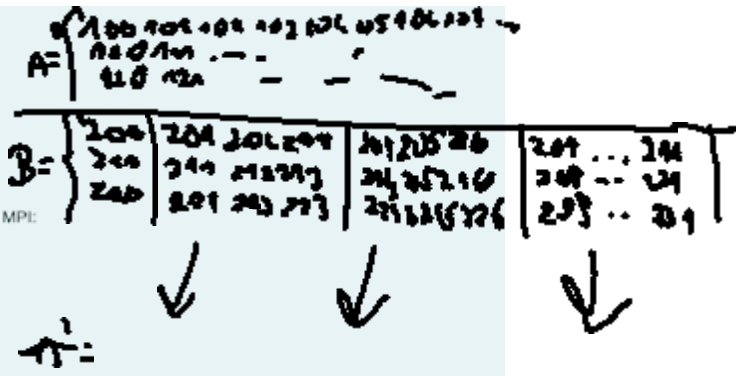
1. Wynik operacji jest zapisany w tablicy B ✗ A

2. Bez zmian w tablicy tej pozostają wyrazy o indeksach mniejszych niż 1 ✓

(w przypadku gdy modyfikowane są wyrazy od samego początku tablicy wpisz 0)

3. Tablica ta dla indeksów od 2 do 7 (np. T[2], T[3], ..., T[7] - jeśli T jest tablicą docelową, w sumie 6 wyrazów) oraz dla poszczególnych procesów ma wartości:

Proces o randze **0**:



6.

Napisz jakie wartości zawiera tablica A dla procesów o **rank** równym: 0, 1, 2, 3 gdzie liczba procesów **size** = 4, po wykonaniu polecenia **MPI_Allgather** dla następujących warunków początkowych:

int A[10] = { 0, 0, 0, 0, 0, 0,0, 0, 0, 0 }; for (i = 0; i < 10; i++) { A[i] = (rank + i) * 2; }

int B[10] = { 0, 0 ,0, 0, 0, 0, 0, 0, 0, 0 }; for (i = 0; i < 10; i++) { B[i] = i; }

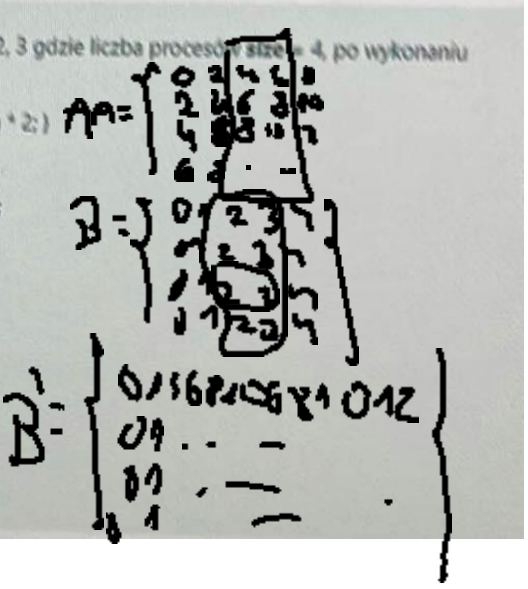
MPI_Allgather(&A[2], 2, MPI_INT, B, 2, MPI_INT, MPI_COMM_WORLD);

// proces 0: A[0] = 0 A[1] = 2

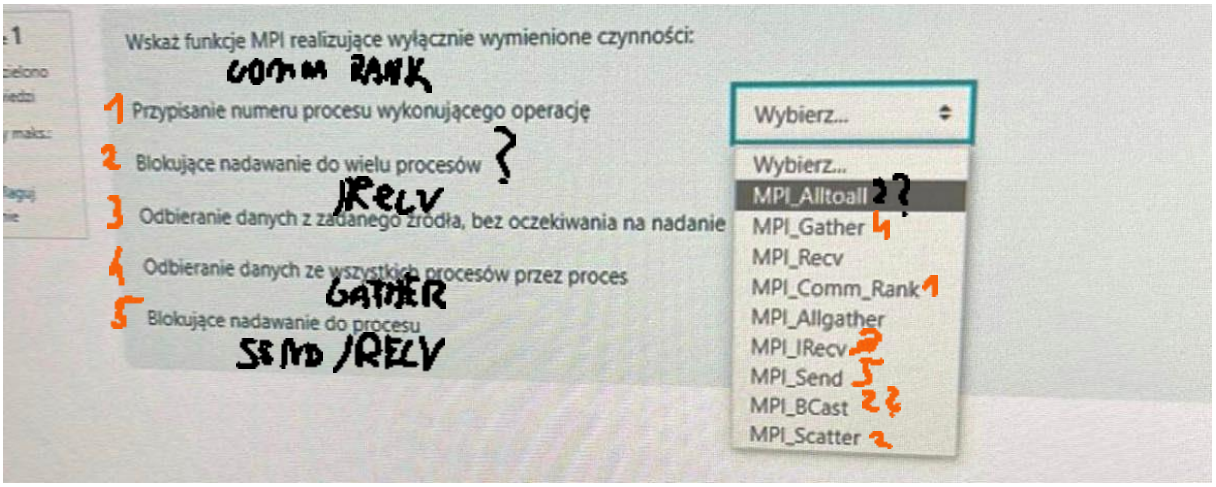
// proces 1: A[0] = 2 A[1] = 4

// proces 2: A[0] = 4 A[1] = 6

// proces 3: A[0] = 6 A[1] = 8



7.



8.

Proszę napisać które iteracje pętli **for** dostaną **3 pierwsze wątki** (wpisz **numer iteracji** lub znak '**-**') w trakcie równoległych obliczeń z wykorzystaniem OpenMP?

static int **N** = **10**; // Liczba iteracji

static int **P** = **4**; // Liczba wątków

```
omp_set_num_threads(P);
#pragma omp parallel for schedule(static,3) default(none) firstprivate(N)
for(int i=0; i<N; i++) {
 printf("W: %d ; I: %d\n",omp_get_thread_num(),i);
```

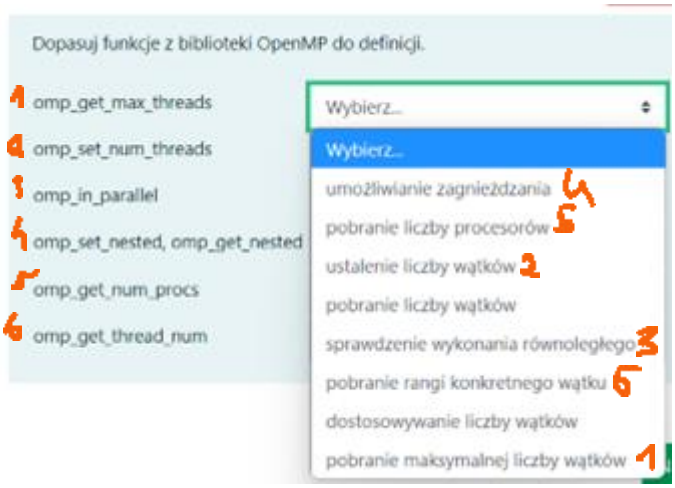
// Wątek 0: 0 ✔ ; 1 ✔ ; 2 ✔ ; - ✔

// Wątek 1: 3 ✔ ; 4 ✔ ; 5 ✔ ; - ✔

// Wątek 2: 6 ✔ ; 7 ✔ ; 8 ✔ ; - ✔

```
}
```

9.



10.

Proszę podać do jakiej sytuacji może dojść podczas współbieżnego wykonania przez wiele wątków (uruchomionych za pomocą pthread_create) poniższej funkcji (gdzie arg jest poprawnie przesłanym wskaźnikiem do zmiennej, a mutex poprawnie zainicjowaną globalną zmienną typu pthread_mutex_t):

```
void thread_fun(void *arg) {
    pthread_mutex_lock(&mutex);
    int *shared = (int*) arg;
    (*shared) += 1;
}
```

- a. Wyścig
- b. Zakleszczenie
- c. Zawsze wykona się poprawnie
- d. Zagłodzenie

11.

Jakie elementy są współdzielone przez wątki?

- a. ciąg rozkazów
- b. zawartość rejestrów
- c. przestrzeń adresowa
- d. stos
- e. deskryptory plików
- f. obszar kodu w przestrzeni adresowej

12.

Proszę podać do jakiej sytuacji może dojść podczas współbieżnego wykonania przez wiele wątków (uruchomionych za pomocą pthread_create) poniższej funkcji (gdzie arg jest poprawnie przesłanym wskaźnikiem do zmiennej, a mutex poprawnie zainicjowaną globalną zmienną typu pthread_mutex_t):

```
void thread_fun(void *arg) {
    pthread_mutex_lock(&mutex);
    int *shared = (int*) arg;
    (*shared) += 1;
    pthread_mutex_unlock(&mutex)
}
```

- a. Zagłodzenie
- b. Wyścig
- c. Zawsze wykona się poprawnie
- d. Zakleszczenie

12.

Zadaniem skryptu jest skompilowanie programu składającego się z następujących plików:

- program.c - plik główny programu
- fun_a.c, fun_a.h - pliku dostarczające funkcjonalność fun_a

Proszę uzupełnić brakujące elementy w skrypcie Makefile:

```
LOADER = gcc
CCOMP = gcc

program: program.o fun_a.o
        ${CCOMP} program.o fun_a.o -o program
program.o : program.c
        $(CC) -c program.c

fun_a.o : fun_a.c fun_a.h
        $(CC) -c fun_a.c
```

13.

Proszę napisać które iteracje pętli **for** dostaną **3 pierwsze wątki** (wpisz **numer iteracji** lub znak '**-**') w trakcie równoległych obliczeń z wykorzystaniem OpenMP?

```
static int N = 15; // Liczba iteracji
static int P = 6; // Liczba wątków
omp_set_num_threads(P);
#pragma omp parallel for schedule(static) default(none) firstprivate(N)
for(int i=0; i<N; i++) {
 printf("W: %d ; I: %d\n",omp_get_thread_num(),i);
```

// Wątek 0: 0 ✓ ; 1 ✗ ; 2 ✗ ; - ✓

// Wątek 1: 3 ✗ ; 4 ✗ ; 5 ✗ ; - ✓

// Wątek 2: 6 ✗ ; 9 ✗ ; 12 ✗ ; - ✓

```
}
```

14.

15.

Proszę dopasować definicje do metod klasy Thread w JAVA :

void join() ✔	oczekiwanie na zakończenie wątku
void start() ✔	polecenie dla JVM rozpoczęcia pracy wątku i wykonania metody run
static void sleep(long millis) ✔	uśpienie bieżącego wątku na określony czas
void interrupt() ✖	natychmiastowe przerwanie działania wątku
void run() ✖	uruchomienie przeciążonej funkcji wykonywania zadań

Twoja odpowiedź jest częściowo poprawna.

Poprawnie wybrałeś 3.
Poprawna odpowiedź to:
void join() → oczekiwanie na zakończenie wątku,

void start() → polecenie dla JVM rozpoczęcia pracy wątku i wykonania metody run,

static void sleep(long millis) → uśpienie bieżącego wątku na określony czas,

void interrupt() → ustawienie sygnalizatora przerwania wątku,

void run() → natychmiastowy powrót, jeśli wątek nie otrzymał w konstruktorze obiektu typu Runnable

16.

Uzupełnij kod odpowiedzialny za tworzenie puli wątków w JAVA (pula o stałej liczbie wątków równej NTHREADS):

```
import java.util.concurrent. ...; //importy odpowiedzialne za obsługę puli wątków
import java.util.concurrent. ...;

public class Simple_test {
private static final int NTHREADS = 10;

public static void main(String[] args) {

Licznik licznik = new Licznik();
[execute] ✖ obslugaPuli= Executors.newFixedThreadPool(NTHREADS);

for (int i = 0; i < 50; i++)
{
Runnable zwiekszaczLicznikow = new ZwiekszaczLicznikow(licznik);
obslugaPuli.[exe] ✖ (zwiekszaczLicznikow);  //zlecenie zadania do realizacji
}
obslugaPuli.[shut] ✖ (); // egzekutor przestaje przyjmować nowe wątki
while (!obslugaPuli.[isTer] ✖ ()) {} // oczekiwanie na zakończenie pracy wątków

 }
}
```

ExecutorService
execute
shutdown
isTerminated

17.

Proszę napisać które iteracje pętli **for** dostaną **3 pierwsze wątki** (wpisz **numer iteracji** lub znak '-') w trakcie równoległych obliczeń z wykorzystaniem OpenMP?

```
static int N = 15; // Liczba iteracji
static int P = 6; // Liczba wątków
omp_set_num_threads(P);
#pragma omp parallel for schedule(static) default(none) firstprivate(N)
for(int i=0; i<N; i++) {
 printf("W: %d ; I: %d\n",omp_get_thread_num(),i);
```

// Wątek 0: 0 ✔ ; 1 ✖ ; 2 ✖ ; - ✔

// Wątek 1: 3 ✖ ; 4 ; 5

// Wątek 2: 6 ✖ ; 7 ; 8

Niepoprawnie
Poprawna odpowiedź to: 2
Punkty: 0,00 z 1,00

```
}
```

18.

Zakładając następujące warunki początkowe:
1. Liczba procesów w ramach komunikatora **MPI_COMM_WORLD**: **size** = 3
2. Identyfikator każdego procesu z zakresu od **0** do **size-1** przechowywany w zmiennej: rank
3. Tablice **A** i **B** zdefiniowane i zainicjowane kodem:

int A[100] = { 0 }; for(i=0; i<100; i++) { A[i] = 100+10*rank+i; }
int B[100] = { 0 }; for(i=0; i<100; i++) { B[i] = 200+10*rank+i; }

proszę uzupełnić poniższe stwierdzenia dotyczące stanu tablic **A** i **B** po wykonaniu polecenia MPI:

MPI_Alltoall(&A[1], 2, MPI_INT, &B[0], 2, MPI_INT, MPI_COMM_WORLD);

(najlepiej rozwiązać zadanie na kartce i następnie wpisać poprawne wartości)

1. Wynik operacji jest zapisany w tablicy B ✔

2. Bez zmian w tablicy tej pozostają wyrazy o indeksach mniejszych niż 0 ✔

(w przypadku gdy modyfikowane są wyrazy od samego początku tablicy wpisz 0)

3. Tablica ta dla indeksów od 1 do 6 (np. T[1], T[2], ..., T[6] - jeśli T jest tablicą docelową, w sumie 6 wyrazów) oraz dla poszczególnych procesów ma wartości:

Proces o randze **0**:

102 ✔ ; 111 ✔ ; 112 ✔ ; 121 ✔ ; 122 ✔ ; 107 ✖ ;

Proces o randze **1**:

104 ✔ ; 113 ✔ ; 114 ✔ ; 123 ✔ ; 124 ✔ ; 107 ✖ ;

Proces o randze **2**:

106 ✔ ; 115 ✔ ; 123 ✖ ; 125 ✔ ; 126 ✔ ; 127 ✖ ;

19.

Uzupełnij kod odpowiedzialny za tworzenie puli wątków w JAVA (pula o stałej liczbie wątków równej NTHREADS):

```
import java.util.concurrent. ...; //importy odpowiedzialne za obsługę puli wątków
import java.util.concurrent. ...;

public class Simple_test {
private static final int NTHREADS = 10;

public static void main(String[] args) {

Licznik licznik = new Licznik();
```

Execute ✖ obslugaPuli= Executors.newFixedThreadPool(NTHREADS);

```
for (int i = 0; i < 50; i++)
{
Runnable zwiekszaczLicznikow = new ZwiekszaczLicznikow(licznik);
```

obslugaPuli. run ✖ (zwiekszaczLicznikow); //zlecenie zadania do realizacji

```
}
```

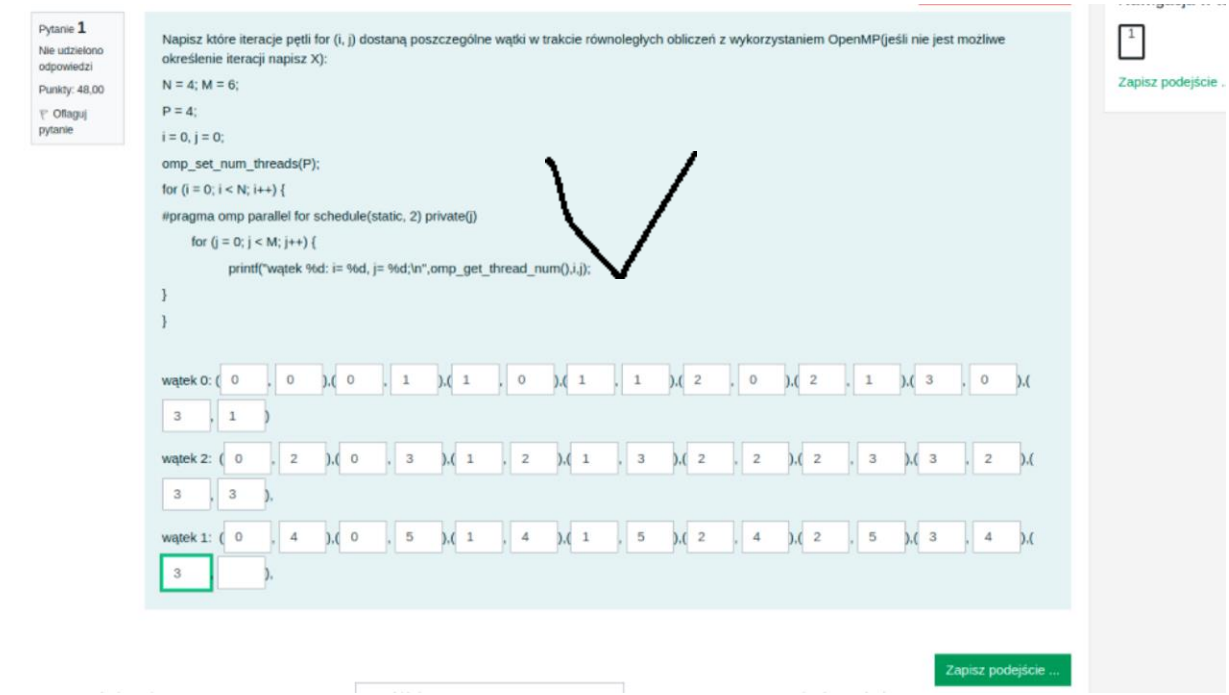
obslugaPuli. run ✖ (); // egzekutor przestaje przyjmować nowe wątki

while (!obslugaPuli. run ✖ ()) {} // oczekiwanie na zakończenie pracy wątków

```
}
}
```

Niepoprawnie
Poprawna odpowiedź to: isTerminated
Punkty: 0,00 z 1,00

20.



21.

Napisz które iteracje pętli for (i, j) dostaną poszczególne wątki w trakcie równoległych obliczeń z wykorzystaniem OpenMP(jeśli nie jest możliwe określenie iteracji napisz X):

```
int N = 9;
int M = 2;
int P = 4;
int i = 0, j = 0;
omp_set_num_threads(P);
#pragma omp parallel for schedule(static, 2) private(j)
for (i = 0; i < N; i++) {
for (j = 0; j < M; j++) {
        printf("wątek %d: i= %d, j= %d;\n",omp_get_thread_num(),i,j);
}
}
```

wątek 0: (0,0),(0,1),(1,0),(1,1),(2,0),(2,1).

wątek 1: (2,0),(2,1),(3,0),(3,1),(-,-),(-,-).

wątek 2: (4,0),(4,1),(5,0),(5,1).

wątek 3: (6,0),(6,1),(7,0),(7,1).

wątek 4: (-,-),(-,-),(-,-),(-,-).

22.

Proszę dokonać dekompozycji pętli w sposób **cykliczny**
dla następujących danych, w miejsca gdzie nie powinno być danych proszę wpisać '-':
A[10] = [7, 4, 9, 1, 4, 5, 0, 5, 8, 3]
LW = 4

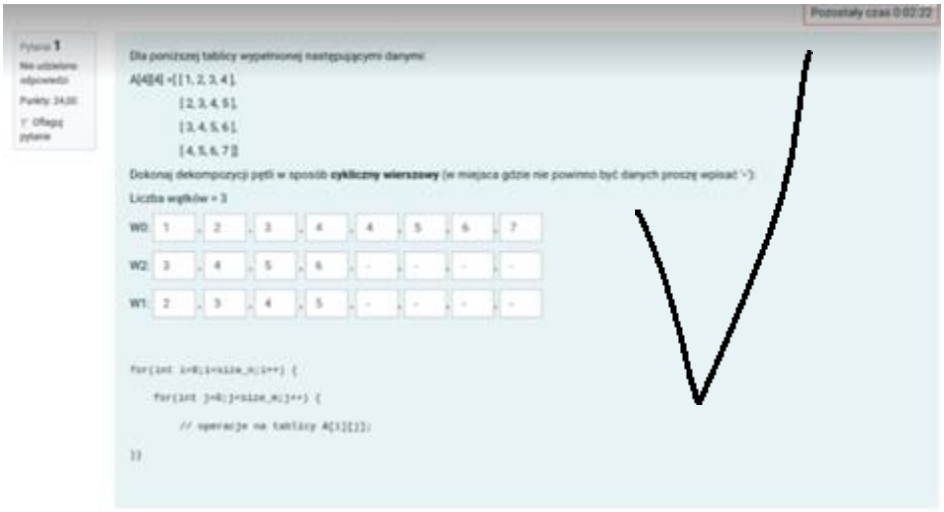
W0: 7, 4, 8, -

W1: 4, 5, 3, -

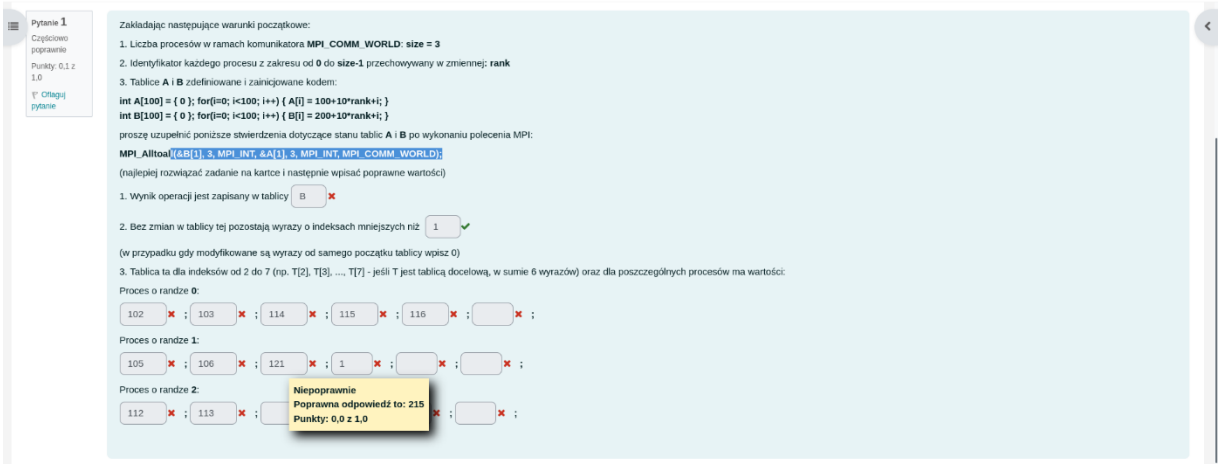
W2: 9, 0, -, -

W3: 1, 5, -

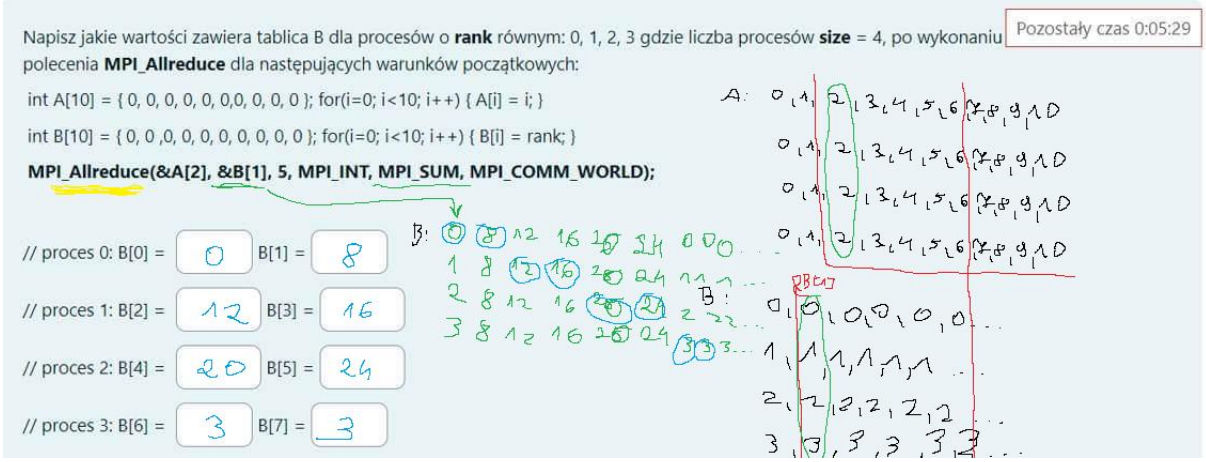
23.


Pozostały czas 0:02:22
Pytanie 1
Dla poniższej tablicy wypełnionej następującymi danymi:
A[4][4] ={[1, 2, 3, 4],
[2, 3, 4, 5],
[3, 4, 5, 6],
[4, 5, 6, 7]}
Dokonaj dekompozycji pętli w sposób cykliczny wierszowy (w miejsca gdzie nie powinno być danych proszę wpisać '-'):
Liczba wątków = 3
W0: 1 2 3 4 4 5 6 7
W2: 3 4 5 6 - - - -
W1: 2 3 4 5 - - - -
for(int i=0;i<size_m;i++) {
for(int j=0;j<size_m;j++) {
// operacje na tablicy A[i][j];
}}

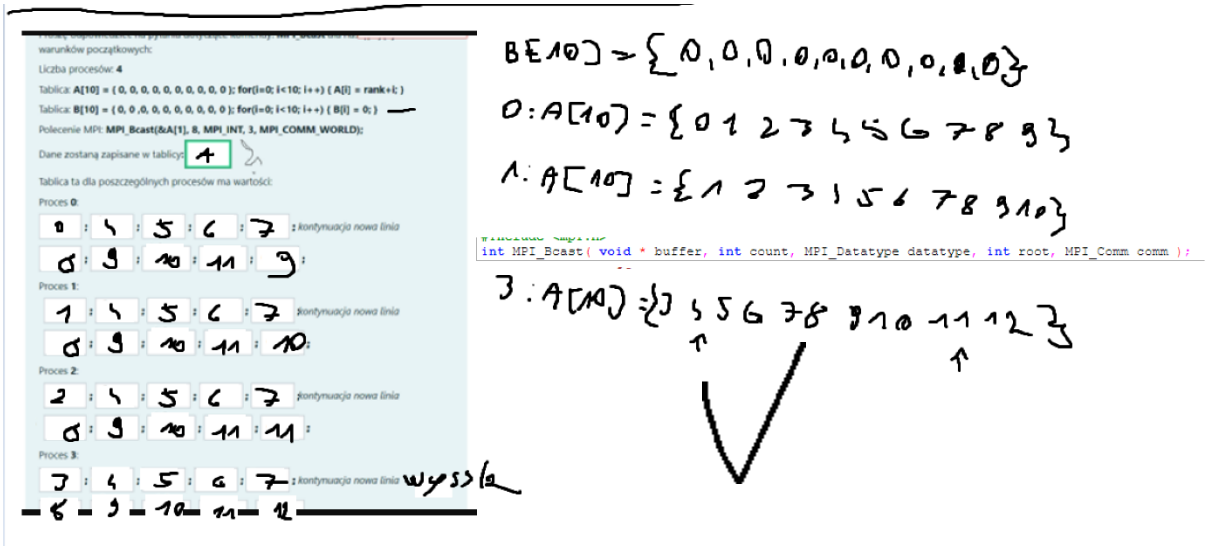
24.


Pytanie 1
Częściowo poprawnie
Punkty: 0,1 z 1,0
Oflaguj pytanie
Zakładając następujące warunki początkowe:
1. Liczba procesów w ramach komunikatora MPI_COMM_WORLD: size = 3
2. Identyfikator każdego procesu z zakresu od 0 do size-1 przechowywany w zmiennej: rank
3. Tablice A i B zdefiniowane i zainicjowane kodem:
int A[100] = { 0 }; for(i=0; i<100; i++) { A[i] = 100+10*rank+i; }
int B[100] = { 0 }; for(i=0; i<100; i++) { B[i] = 200+10*rank+i; }
proszę uzupełnić poniższe stwierdzenia dotyczące stanu tablic A i B po wykonaniu polecenia MPI:
MPI_Alltoall(&B[1], 3, MPI_INT, &A[1], 3, MPI_INT, MPI_COMM_WORLD);
(najlepiej rozwiązać zadanie na kartce i następnie wpisać poprawne wartości)
1. Wynik operacji jest zapisany w tablicy B
2. Bez zmian w tablicy tej pozostają wyrazy o indeksach mniejszych niż 1
(w przypadku gdy modyfikowane są wyrazy od samego początku tablicy wpisz 0)
3. Tablica ta dla indeksów od 2 do 7 (np. T[2], T[3], ..., T[7] - jeśli T jest tablicą docelową, w sumie 6 wyrazów) oraz dla poszczególnych procesów ma wartości:
Proces o randze 0:
102 ; 103 ; 114 ; 115 ; 116 ; ;
Proces o randze 1:
105 ; 106 ; 121 ; 1 ; ; ;
Proces o randze 2:
112 ; 113 ; ;
Niepoprawnie
Poprawna odpowiedź to: 215
Punkty: 0,0 z 1,0

25.


Napisz jakie wartości zawiera tablica B dla procesów o rank równym: 0, 1, 2, 3 gdzie liczba procesów size = 4, po wykonaniu
Pozostały czas 0:05:29
polecenia MPI_Allreduce dla następujących warunków początkowych:
int A[10] = { 0, 0, 0, 0, 0, 0,0, 0, 0, 0 }; for(i=0; i<10; i++) { A[i] = i; }
int B[10] = { 0, 0 ,0, 0, 0, 0, 0, 0, 0, 0 }; for(i=0; i<10; i++) { B[i] = rank; }
MPI_Allreduce(&A[2], &B[1], 5, MPI_INT, MPI_SUM, MPI_COMM_WORLD);
// proces 0: B[0] = 0 B[1] = 8
// proces 1: B[2] = 12 B[3] = 16
// proces 2: B[4] = 20 B[5] = 24
// proces 3: B[6] = 3 B[7] = 3

26.



27.

Napisz jakie wartości zawiera tablica B dla procesów o **rank** równym: 0, 1, 2, 3 gdzie liczba procesów **size** = 4, po wykonaniu polecenia **MPI_Allreduce** dla następujących warunków początkowych:

Pozostały czas 0:05:29

int A[10] = { 0, 0, 0, 0, 0, 0,0, 0, 0, 0 }; for(i=0; i<10; i++) { A[i] = i; }

int B[10] = { 0, 0 ,0, 0, 0, 0, 0, 0, 0, 0 }; for(i=0; i<10; i++) { B[i] = rank; }

MPI_Allreduce(&A[2], &B[1], 5, MPI_INT, MPI_SUM, MPI_COMM_WORLD);

// proces 0: B[0] = [ ] B[1] = [ ]

// proces 1: B[2] = [ ] B[3] = [ ]

// proces 2: B[4] = [ ] B[5] = [ ]

// proces 3: B[6] = [ ] B[7] = [ ]

28.

[redacted]

Zakładając następujące warunki początkowe:

1. Liczba procesów w ramach komunikatora **MPI_COMM_WORLD**: **size = 3**
2. Identyfikator każdego procesu z zakresu od **0** do **size-1** przechowywany w zmiennej: **rank**
3. Tablice **A** i **B** zdefiniowane i zainicjowane kodem:

A[100] = { 0 }; for(i=0; i<100; i++) { A[i] = 100*rank+i; }

B[100] = { 0 }; for(i=0; i<100; i++) { B[i] = rank; }

proszę uzupełnić poniższe stwierdzenia dotyczące stanu tablic **A** i **B** po wykonaniu polecenia MPI:

MPI_Alltoall(&A[1], 2, MPI_INT, &B[0], 2, MPI_INT, MPI_COMM_WORLD);

(najlepiej rozwiązać zadanie na kartce i następnie wpisać poprawne wartości)

1. Wynik operacji jest zapisany w tablicy B ✔

2. Bez zmian w tablicy tej pozostają wyrazy o indeksach mniejszych niż 1 ✖

(w przypadku gdy modyfikowane są wyrazy od samego początku tablicy wpisz 0)

2. Tablica ta dla indeksów od 3 do 8 (A[3], A[4], ..., A[8], w sumie 6 wyrazów) oraz dla poszczególnych procesów ma wartości:

Proces o randze **0**:

100 ✖ ; 100 ✖ ; 100 ✖ ; 100 ✖ ; 100 ✖ ; 100 ✖ ;

Proces o

Niepoprawnie
Poprawna odpowiedź to: 102
Ocena: 0,00 z 1,00

100 ✖ ; 100 ✖ ; 100 ✖ ; 100 ✖ ;

Proces o randze **2**:

100 ✖ ; 100 ✖ ; 100 ✖ ; 100 ✖ ; 100 ✖ ; 100 ✖ ;

[redacted]

29.

warunków początkowych:

Liczba procesów: **4**

Tablica: **A[10] = { 0, 0, 0, 0, 0, 0, 0, 0, 0, 0 }; for(i=0; i<10; i++) { A[i] = rank+i; }**

Tablica: **B[10] = { 0, 0 ,0, 0, 0, 0, 0, 0, 0, 0 }; for(i=0; i<10; i++) { B[i] = 0; }**

Polecenie MPI: **MPI_Bcast(&A[1], 8, MPI_INT, 3, MPI_COMM_WORLD);**

Dane zostaną zapisane w tablicy: A

Tablica ta dla poszczególnych procesów ma wartości:

Proces **0**:

0 ; 4 ; 5 ; 6 ; 7 ; *kontynuacja nowa linia*

8 ; 9 ; 10 ; 11 ; 9 ;

Proces **1**:

1 ; 4 ; 5 ; 6 ; 7 ; *kontynuacja nowa linia*

8 ; 9 ; 10 ; 11 ; 10 ;

Proces **2**:

2 ; 4 ; 5 ; 6 ; 7 ; *kontynuacja nowa linia*

8 ; 9 ; 10 ; 11 ; 11 ;

Proces **3**:

3 ; 4 ; 5 ; 6 ; 7 ; *kontynuacja nowa linia* root

8 9 10 11 12

30.