PRZETWARZANIE RÓWNOLEGŁE

ZADANIA

Z1. Uruchom i przeanalizuj poniższy program w języku Python:

```
def PUT():
    global S
    x = S
    x+= 50
    S = x

def ECDL():
    global S
    x = S
    x+= 70
    S = x

S= 7
PUT()
ECDL()
print("S=", S)
```

Z2. Uruchom i przeanalizuj poniższy program w języku C:

```
#include <stdio.h>

int S;

void PUT(){
  int x;
  x = S;
  x+= 50;
  S = x; }

void ECDL(){
  int x;
  x = S;
  x+= 70;
  S = x; }

int main(void){
  S= 7;
  PUT();
  ECDL();
  printf("S=%d\n",S);}
```

Z3. Uruchom i przeanalizuj poniższy program w języku Python:

```
import threading

def PUT():
    global S
    x = S
    x+= 50
    S = x

def ECDL():
    global S
    x = S
    x+= 70
    S = x

S= 7
PUTthread = threading.Thread(target=PUT)
ECDLthread= threading.Thread(target=ECDL)
PUTthread.start()
ECDLthread.start()
PUTthread.join()
ECDLthread.join()
print("S= ", S)
```

Z4. Uruchom i przeanalizuj poniższy program w języku C:

```
#include <pthread.h>
#include <stdio.h>

int S;

void *PUT(void* arg){
  int x;
  x= S; x+= 50; S= x;
  return NULL; }

void *ECDL(void* arg){
  int x;
  x= S; x+= 70; S= x;
  return NULL; }

int main(void){
  pthread_t PUT_h, ECDL_h;
  S= 7;
  pthread_create(&PUT_h, NULL, PUT, NULL);
  pthread_create(&ECDL_h, NULL, ECDL, NULL);
  pthread_join(PUT_h, NULL);
  pthread_join(ECDL_h, NULL);
  printf("S= %d\n", S); }
```

Z5. Uruchom i przeanalizuj poniższy program w języku Python:

```
import threading

def PUT():
    global S
    x = S
    x+= 50
    S = x

def ECDL():
    global S
    x = S
    x+= 70
    S = x

def run2(f1, f2):
    a = threading.Thread(target= f1)
    b = threading.Thread(target= f2)
    a.start()
    b.start()
    a.join()
    b.join()

S= 7
run2(PUT, ECDL)
print("S= ", S)
```

Z6. Uruchom i przeanalizuj poniższy program w języku C:

```
#include <pthread.h>
#include <stdio.h>

int S;

void *PUT(void* arg){
  int x;
  x= S; x+= 50; S= x;
  return NULL; }

void *ECDL(void* arg){
  int x;
  x= S; x+= 70; S= x;
  return NULL; }

void run2(void*(*f1)(void* arg), void*(*f2)(void* arg)){
  pthread_t h1, h2;
  pthread_create(&h1, NULL, f1, NULL);
  pthread_create(&h2, NULL, f2, NULL);
  pthread_join(h1, NULL);
  pthread_join(h2, NULL); }

int main(void){
  S= 7;
  run2(PUT, ECDL);
  printf("S= %d\n", S); }
```

Z7. Uruchom i przeanalizuj poniższy program w języku Python:

```
import threading
import time

def PUT():
    global S
    x = S
    time.sleep(2)
    x+= 50
    S = x

def ECDL():
    global S
    x = S
    x+= 70
    S = x

def run2(f1, f2):
    a = threading.Thread(target= f1)
    b = threading.Thread(target= f2)
    a.start()
    b.start()
    a.join()
    b.join()

S= 7
run2(PUT, ECDL)
print("S= ", S)
```

Z8. Uruchom i przeanalizuj poniższy program w języku Python:

```
import threading
import time

def PUT():
    global S
    x = S
    time.sleep(2)
    x+= 50
    S = x

def ECDL():
    global S
    x = S
    x+= 70
    S = x

def run2(f1, f2):
    a = threading.Thread(target= f1)
    b = threading.Thread(target= f2)
    a.start()
    b.start()
    a.join()
    b.join()

S= 7
run2(PUT, ECDL)
print("S= ", S)
```

Z9. Uruchom i przeanalizuj poniższy program w języku C:

```
#include <unistd.h>
#include <pthread.h>
#include <stdio.h>

int S;

void *PUT(void* arg){
  int x;
  x= S; sleep(2); x+= 50; S= x;
  return NULL; }
void *ECDL(void* arg){
  int x;
  x= S; x+= 70; S= x;
  return NULL; }

void run2(void*(*f1)(void* arg), void*(*f2)(void* arg)){
  pthread_t h1, h2;
  pthread_create(&h1, NULL, f1, NULL);
  pthread_create(&h2, NULL, f2, NULL);
  pthread_join(h1, NULL);
  pthread_join(h2, NULL);}

int main(void){
  S= 7;
  run2(ECDL, PUT);
  printf("S= %d\n", S); }
```

Z10. Uruchom i przeanalizuj poniższy program w języku Python:

```
import threading
import time

def PUT():
    global S
    mutex.acquire()
    x = S
    time.sleep(2)
    x+= 50
    S = x
    mutex.release()

def ECDL():
    global S
    mutex.acquire()
    x = S
    x+= 70
    S = x
    mutex.release()

def run2(f1, f2):
    a = threading.Thread(target= f1)
    b = threading.Thread(target= f2)
    a.start()
    b.start()
    a.join()
    b.join()

S= 7
mutex = threading.Semaphore()
run2(PUT, ECDL)
print("S= ", S)
```

Z11. Uruchom i przeanalizuj poniższy program w języku C:

```
#include <unistd.h>
#include <pthread.h>
#include <semaphore.h>
#include <stdio.h>

int S; sem_t mutex;

void *PUT(void* arg){
  int x;
  sem_wait(&mutex);
  x= S; sleep(2); x+= 50; S= x;
  sem_post(&mutex);
  return NULL; }

void *ECDL(void* arg){
  int x;
  sem_wait(&mutex);
  x= S; x+= 70; S= x;
  sem_post(&mutex);
  return NULL; }

void run2(void*(*f1)(void* arg), void*(*f2)(void* arg)){
  pthread_t h1, h2;
  pthread_create(&h1, NULL, f1, NULL);
  pthread_create(&h2, NULL, f2, NULL);
  pthread_join(h1, NULL);
  pthread_join(h2, NULL);}

int main(void){
  S= 7;
  sem_init(&mutex, 0, 1);
  run2(PUT, ECDL);
  printf("S= %d\n", S); }
```

Z12. Uruchom i przeanalizuj poniższy program w języku Python:

```
import threading
import time

def produce():
    print('producing')
    time.sleep(1)
    print('done')

def insert():
    global storage
    storage = storage + 1

def consume():
    print('consuming')
    time.sleep(1)
    print('done')

def take():
    global storage
    storage = storage - 1


def producer():
    i = 0
    while i < 2:
        produce()
        Empty.acquire()
        mutex.acquire()
        insert()
        mutex.release()
        Full.release()
        i = i + 1

def consumer():
    j = 0
    while j < 2:
        Full.acquire()
        mutex.acquire()
        take()
        mutex.release()
        Empty.release()
        consume()
        j = j + 1

def run2(f1, f2):
    a = threading.Thread(target= f1)
    b = threading.Thread(target= f2)
    a.start()
    b.start()
    a.join()
    b.join()

storage= 0
Empty= threading.Semaphore(10)
Full = threading.Semaphore(0)
mutex= threading.Semaphore()
run2(producer, consumer)
print("storage= ", storage)
```

Z13. Uruchom i przeanalizuj poniższy program w języku Python:

```
import threading
import time

def r_using():
    print('r_using')
    time.sleep(1)
    print('done')

def r_reading():
    print('r_reading')
    time.sleep(2)
    print('done')

def a_thinking():
    print('a_thinking')
    time.sleep(1)
    print('done')

def a_writing():
    print('a_writing')
    time.sleep(3)
    print('done')
def reader():
    global numRe
    i = 0
    while i < 2:
        mutex.acquire()
        numRe+= 1
        if numRe == 1:
            db.acquire()
        mutex.release()
        r_reading()
        mutex.acquire()
        numRe-= 1
        if numRe == 0:
            db.release()
        mutex.release()
        r_using()
        i = i + 1

def writer():
    j = 0
    while j < 2:
        a_thinking()
        db.acquire()
        a_writing()
        db.release()
        j = j + 1

def run4(f1, f2):
    a = threading.Thread(target= f1)
    b = threading.Thread(target= f1)
    c = threading.Thread(target= f1)
    d = threading.Thread(target= f2)
    a.start()
    b.start()
    c.start()
    d.start()
    a.join()
    b.join()
    c.join()
    d.join()

numRe = 0
db= threading.Semaphore()
mutex= threading.Semaphore()
run4(reader, writer)
```

Z14. Uruchom i przeanalizuj poniższy program w języku Python:

```
import threading

def funA():
    global X
    global Y
    muteX.acquire()
    muteY.acquire()
    X= X+2
    Y= Y+2
    print("A:", X, Y)
    muteY.release()
    muteX.release()

def funB():
    global X
    global Y
    muteY.acquire()
    muteX.acquire()
    X= X+2
    Y= Y+2
    print("B:", X, Y)
    muteX.release()
    muteY.release()

def run2(f1, f2):
    a = threading.Thread(target= f1)
    b = threading.Thread(target= f2)
    a.start()
    b.start()
    a.join()
    b.join()

X= 2
muteX= threading.Semaphore()
Y= 2
muteY= threading.Semaphore()
run2(funA, funB)
```

Z15. Uruchom i przeanalizuj poniższy program w języku Python:

```
import threading
import time

def funA():
    global X
    global Y
    muteX.acquire()
    time.sleep(1)
    muteY.acquire()
    X= X+2
    Y= Y+2
    print("A:", X, Y)
    muteY.release()
    muteX.release()

def funB():
    global X
    global Y
    muteY.acquire()
    muteX.acquire()
    X= X+2
    Y= Y+2
    print("B:", X, Y)
    muteX.release()
    muteY.release()

def run2(f1, f2):
    a = threading.Thread(target= f1)
    b = threading.Thread(target= f2)
    a.start()
    b.start()
    a.join()
    b.join()

X= 2
muteX= threading.Semaphore()
Y= 2
muteY= threading.Semaphore()
run2(funA, funB)
```

Z16. Uruchom i przeanalizuj poniższy program w języku Python:

```
import threading
import time

def funA():
    global X
    global Y
    mutex.acquire()
    time.sleep(1)
    X= X+2
    Y= Y+2
    print("A:", X, Y)
    mutex.release()

def funB():
    global X
    global Y
    mutex.acquire()
    X= X+2
    Y= Y+2
    print("B:", X, Y)
    mutex.release()

def run2(f1, f2):
    a = threading.Thread(target= f1)
    b = threading.Thread(target= f2)
    a.start()
    b.start()
    a.join()
    b.join()

X= 2
Y= 2
mutex= threading.Semaphore()
run2(funA, funB)
```

Z17. Uruchom i przeanalizuj poniższy program w języku Python:

```
import threading
import time

def funA():
    global X
    global Y
    muteX.acquire()
    time.sleep(1)
    muteY.acquire()
    X= X+2
    Y= Y+2
    print("A:", X, Y)
    muteY.release()
    muteX.release()

def funB():
    global X
    global Y
    muteX.acquire()
    muteY.acquire()
    X= X+2
    Y= Y+2
    print("B:", X, Y)
    muteY.release()
    muteX.release()

def run2(f1, f2):
    a = threading.Thread(target= f1)
    b = threading.Thread(target= f2)
    a.start()
    b.start()
    a.join()
    b.join()

X= 2
muteX= threading.Semaphore()
Y= 2
muteY= threading.Semaphore()
run2(funA, funB)
```