

|                    |                          |
|--------------------|--------------------------|
| <b>NAMA</b>        | <b>INGWE IANNICO</b>     |
| <b>NBI</b>         | <b>1461600214</b>        |
| <b>KELAS</b>       | <b>R</b>                 |
| <b>MATA KULIAH</b> | <b>KOMPUTASI PARALEL</b> |

## JAWABAN NO 1.

```

print ("\n\nINGWE IANNICO")
print ("1461600214")

import multiprocessing
import time

def genap11():
    for i in range(11,21,1):
        if (i % 2 == 0):
            print ('Proses1.a : ',i)
            time.sleep(1)

def genappr():
    for j in range (1,11,1):
        if (j % 2 == 0):
            print('Proses1.b : ', j)
            time.sleep(1.2)

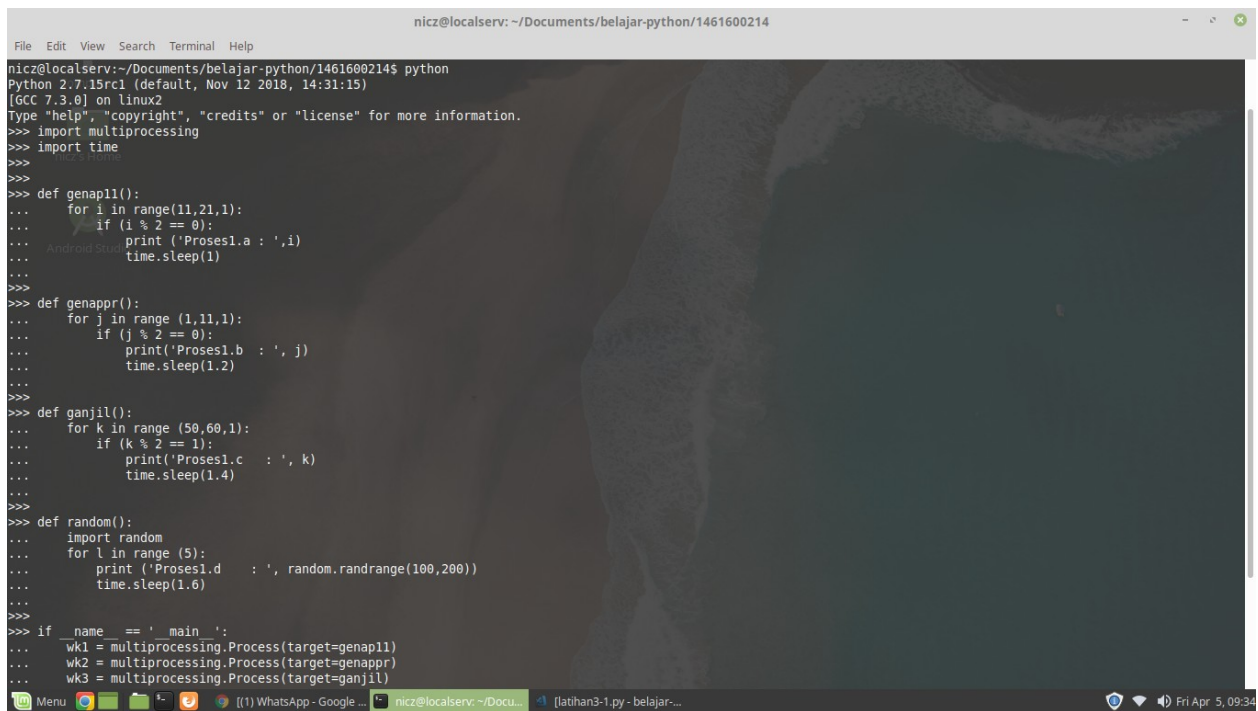
def ganjil():
    for k in range (50,60,1):
        if (k % 2 == 1):
            print('Proses1.c : ', k)
            time.sleep(1.4)

def random():
    import random
    for l in range (5):
        print ('Proses1.d : ', random.randrange(100,200))
        time.sleep(1.6)

if __name__ == '__main__':
    wk1 = multiprocessing.Process(target=genap11)

```

```
wk2 = multiprocessing.Process(target=genappr)
wk3 = multiprocessing.Process(target=ganjil)
wk4 = multiprocessing.Process(target=random)
wk1.start()
wk2.start()
wk3.start()
wk4.start()
wk1.join()
wk2.join()
wk3.join()
wk4.join()
```



```
nicz@localserv: ~/Documents/belajar-python/1461600214
File Edit View Search Terminal Help
nicz@localserv:~/Documents/belajar-python/1461600214$ python
Python 2.7.15rc1 (default, Nov 12 2018, 14:31:15)
[GCC 7.3.0] on linux2
Type "help()", "copyright()", "credits()" or "license()" for more information.
>>> import multiprocessing
>>> import time
>>>
>>> def genappr1():
...     for i in range(11,21,1):
...         if (i % 2 == 0):
...             print ('Proses1.a : ',i)
...             time.sleep(1)
...
>>> def genappr():
...     for j in range (1,11,1):
...         if (j % 2 == 0):
...             print('Proses1.b : ', j)
...             time.sleep(1.2)
...
>>> def ganjil():
...     for k in range (50,60,1):
...         if (k % 2 == 1):
...             print('Proses1.c : ', k)
...             time.sleep(1.4)
...
>>> def random():
...     import random
...     for l in range (5):
...         print ('Proses1.d : ', random.randrange(100,200))
...         time.sleep(1.6)
...
>>>
>>> if __name__ == '__main__':
...     wk1 = multiprocessing.Process(target=genappr1)
...     wk2 = multiprocessing.Process(target=genappr)
...     wk3 = multiprocessing.Process(target=ganjil)
```

```
nicz@localserv: ~/Documents/belajar-python/1461600214
File Edit View Search Terminal Help
...
for i in range(11,21,1):
    if (i % 2 == 0):
        print ('Proses1.a : ',i)
        time.sleep(1)
...
def genappr():
    for j in range (1,11,1):
        if (j % 2 == 0):
            print('Proses1.b : ', j)
            time.sleep(1.2)
...
def ganjil():
    for k in range (50,60,1):
        if (k % 2 == 1):
            print('Proses1.c : ', k)
            time.sleep(1.4)
...
def random():
    import random
    for l in range (5):
        print ('Proses1.d : ', random.randrange(100,200))
        time.sleep(1.6)
...
if __name__ == '__main__':
    wk1 = multiprocessing.Process(target=genappr)
    wk2 = multiprocessing.Process(target=genappr)
    wk3 = multiprocessing.Process(target=ganjil)
    wk4 = multiprocessing.Process(target=random)
    wk1.start()
    wk2.start()
    wk3.start()
    wk4.start()
    wk1.join()
    wk2.join()
    wk3.join()
    wk4.join()
...
Menu [Icons] nicz@localserv: ~/Docu... [latihan3-1.py - belajar...
```

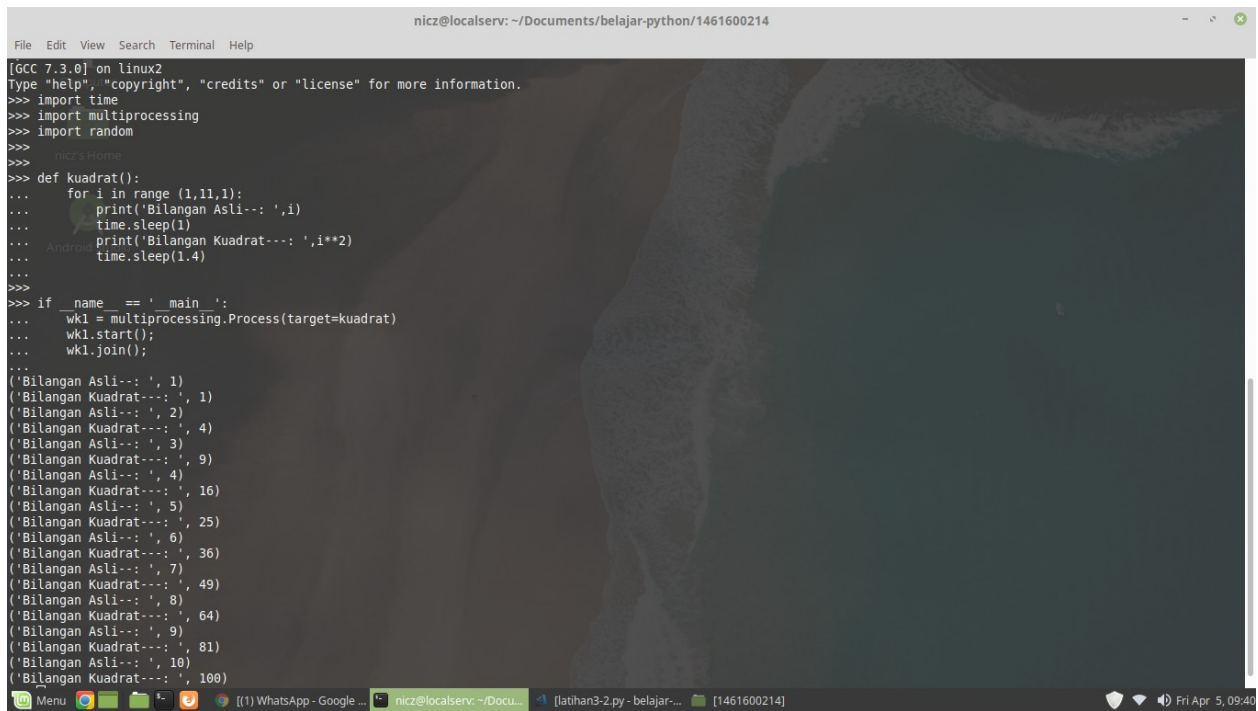
```
nicz@localserv: ~/Documents/belajar-python/1461600214
File Edit View Search Terminal Help
...
    print ('Proses1.d : ', random.randrange(100,200))
    time.sleep(1.6)
...
if __name__ == '__main__':
    wk1 = multiprocessing.Process(target=genappr)
    wk2 = multiprocessing.Process(target=genappr)
    wk3 = multiprocessing.Process(target=ganjil)
    wk4 = multiprocessing.Process(target=random)
    wk1.start()
    wk2.start()
    wk3.start()
    wk4.start()
    wk1.join()
    wk2.join()
    wk3.join()
    wk4.join()
...
('Proses1.a : ', 12)
('Proses1.b : ', 2)
('Proses1.c : ', 51)
('Proses1.d : ', 121)
('Proses1.a : ', 14)
('Proses1.b : ', 4)
('Proses1.c : ', 53)
('Proses1.d : ', 146)
('Proses1.a : ', 16)
('Proses1.b : ', 6)
('Proses1.c : ', 55)
('Proses1.a : ', 18)
('Proses1.d : ', 115)
('Proses1.b : ', 8)
('Proses1.a : ', 20)
('Proses1.c : ', 57)
('Proses1.b : ', 10)
('Proses1.d : ', 187)
('Proses1.c : ', 59)
('Proses1.d : ', 188)
...
Menu [Icons] nicz@localserv: ~/Docu... [latihan3-1.py - belajar...
```

## JAWABAN NO 2.

```
import time
import multiprocessing
import random

def kuadrat():
    for i in range (1,11,1):
        print('Bilangan Asli--: ',i)
        time.sleep(1)
        print('Bilangan Kuadrat---: ',i**2)
        time.sleep(1.4)

if __name__ == '__main__':
    wk1 = multiprocessing.Process(target=kuadrat)
    wk1.start();
    wk1.join();
```



```
nicz@localserv: ~/Documents/belajar-python/1461600214
File Edit View Search Terminal Help

[GCC 7.3.0] on linux2
Type "help", "copyright", "credits" or "license" for more information.
>>> import time
>>> import multiprocessing
>>> import random
>>>
>>> def kuadrat():
...     for i in range (1,11,1):
...         print('Bilangan Asli--: ',i)
...         time.sleep(1)
...         print('Bilangan Kuadrat---: ',i**2)
...         time.sleep(1.4)
...
>>>
>>> if __name__ == '__main__':
...     wk1 = multiprocessing.Process(target=kuadrat)
...     wk1.start();
...     wk1.join();
...
('Bilangan Asli--: ', 1)
('Bilangan Kuadrat---: ', 1)
('Bilangan Asli--: ', 2)
('Bilangan Kuadrat---: ', 4)
('Bilangan Asli--: ', 3)
('Bilangan Kuadrat---: ', 9)
('Bilangan Asli--: ', 4)
('Bilangan Kuadrat---: ', 16)
('Bilangan Asli--: ', 5)
('Bilangan Kuadrat---: ', 25)
('Bilangan Asli--: ', 6)
('Bilangan Kuadrat---: ', 36)
('Bilangan Asli--: ', 7)
('Bilangan Kuadrat---: ', 49)
('Bilangan Asli--: ', 8)
('Bilangan Kuadrat---: ', 64)
('Bilangan Asli--: ', 9)
('Bilangan Kuadrat---: ', 81)
('Bilangan Asli--: ', 10)
('Bilangan Kuadrat---: ', 100)
```

### **JAWABAN NO 3.**

#### **MODEL MEMORI YANG DIBAGI**

adalah objek fisik yang memungkinkan analisis atau pengamatannya eksplorasi objek atau proses nyata lainnya. Model mewakili objek yang dieksplorasi dengan cara yang disederhanakan dengan hanya memperhitungkan fitur-fitur dasarnya. Karena penyederhanaan, model lebih mudah dianalisis daripada objek nyata yang sesuai. model komputer yang memungkinkan studi tentang proses komputasi yang dilakukan di dalam komputer tersebut. Model-model, yang disebut model perhitungan, sangat membantu ketika menganalisis dan merancang algoritma, serta dalam menentukan metrik kinerja yang digunakan untuk evaluasi algoritma

Model perhitungan tidak boleh dikaitkan dengan arsitektur komputer tertentu, atau dengan kelas arsitektur seperti itu. Dengan kata lain, independensi mereka dari perangkat keras sangat penting. Fitur penting lainnya adalah fleksibilitas yang memastikan algoritma yang dikembangkan mengadopsi model-model ini dapat diimplementasikan dan dijalankan komputer dengan arsitektur yang berbeda. Ini sangat penting dalam bidang komputasi paralel di mana keragaman arsitektur tinggi. Sebagai hasil dari keanekaragaman ini beberapa model telah maju. Sayangnya, karena jumlahnya relatif besar persyaratan bahwa model harus memenuhi, sebagian dalam konflik satu sama lain, tidak ada model yang dikembangkan sejauh ini telah menjadi model paralel yang diterima secara umum komputasi. Yang sering digunakan adalah model memori bersama (atau paralel acak model mesin akses, PRAM) dan model jaringan. Mereka sesuai dengan perhitungan paralel yang dilakukan dengan menggunakan memori bersama dan dengan mengirim pesan melalui beberapa jaringan komunikasi. Sebelum membahas model-model ini, kami akan hadir model komputasi berurutan yang mendasari model PRAM

#### **MODEL RAM**

Model perhitungan sekuensial yang diterima secara luas adalah mesin dengan acak akses memori (RAM). Model1 terdiri dari prosesor dan memori yang mengandung jumlah sel  $M_i$  yang berpotensi tak terbatas untuk  $i = 1, 2, 3, \dots$ . Di setiap sel memori yang diidentifikasi oleh alamatnya  $i$ , nilai terbatas yang dinyatakan dalam biner, mungkin sangat besar, dapat disimpan. Model ini mengasumsikan bahwa waktu untuk membaca (menulis) nilai dari

(dalam) sebuah sel  $M_i$  adalah konstan dan sama dengan satuan waktu, terlepas dari alamat sel. Konstan ini waktu akses adalah karakteristik untuk akses acak, sebagai lawan dari akses berurutan ke sel-sel kaset di mesin Turing. Prosesor RAM dapat menjalankan instruksi dari beberapa daftar terbatas yang mencakup instruksi yang mirip dengan yang diimplementasikan dalam prosesor modern. Nilainya meningkat oleh 1 setelah pelaksanaan instruksi yang berbeda dari JUMP, JPOS, JZERO dan JNEG. Dalam hal instruksi ini, register L dapat diatur ke nilai  $i$  (the argumen kedua dari suatu instruksi), atau meningkat sebesar 1. Prosesor dikendalikan oleh program RAM, yang merupakan urutan instruksi yang mengimplementasikan algoritma yang diberikan. Program tidak disimpan dalam memori, tetapi dalam unit kontrol model. Ini memastikan bahwa program tidak dapat dimodifikasi selama pelaksanaannya. Saat menganalisis RAM program, mudah untuk mengasumsikan bahwa suatu program terdiri dari sejumlah langkah komputasi, di mana setiap langkah dibagi menjadi tiga fase.

- fase pertama prosesor mengambil operand dari sel memori ke register aritmatika
- fase kedua ia melakukan operasi aritmatika atau logis (Boolean) pada isi register aritmatika, operasi kontrol, operasi I / O dll, dan dalam
- fase ketiga ini menyimpan hasil operasi di sel alamat yang ditunjukkan.

Beberapa fase dari langkah komputasi mungkin kosong. Diasumsikan bahwa waktu eksekusi adalah sama untuk setiap langkah komputasi, dan sama dengan satuan waktu. Nilai yang dihitung dari polinomial muncul di sel  $M_1$  Program RAM yang diungkapkan oleh instruksi prosesor dirinci dan ditutup ke bahasa mesin. Akibatnya, mereka nyaris tidak terbaca. Karena itu untuk meningkatkan keterbacaan mereka biasanya disajikan pada tingkat abstraksi yang lebih tinggi.

## **MODEL JARINGAN**

Model jaringan terdiri dari sejumlah prosesor yang beroperasi sesuai dengan model RAM. Untuk bertukar data, prosesor dihubungkan oleh saluran komunikasi dua arah, atau tautan, yang membentuk jaringan interkoneksi tidak ada memori bersama dalam model. Prosesor dapat beroperasi baik secara sinkron atau asinkron, tetapi biasanya mode asinkron diasumsikan. Prosesor berkomunikasi dan mengoordinasikan kegiatan mereka dengan mengirim pesan melalui tautan jaringan interkoneksi. Transfer pesan

dilakukan dengan prosedur routing yang berjalan di semua prosesor dan bekerja sama dengan masing-masing

lainnya.

Model jaringan mendefinisikan cara prosesor terhubung satu sama lain, yang disebut topologi jaringan interkoneksi. Ini dapat dijelaskan dengan cara grafik yang simpulnya mewakili prosesor, dan tepinya mewakili tautan dua arah antara prosesor. Grafik dapat menjadi diagram alternatif dari model jaringan. Ada banyak topologi jaringan, yang dianalisis dengan mempertimbangkan berbagai parameter yang memungkinkan kami menilai kesesuaian setiap topologi untuk komputasi paralel. Parameter berikut untuk mengevaluasi topologi dibedakan: diameter, derajat, lebar pembelahan, konektivitas tepi, dan biaya. Diameter jaringan adalah jarak maksimum yang diukur dengan jumlah tautan (tepi) antara dua prosesor atau simpul (simpul) dari suatu jaringan. Jaringan adalah semakin baik semakin kecil diameter yang dimilikinya, karena dalam kasus terburuk suatu pesan harus dialihkan melalui jumlah tautan sama dengan diameter jaringan. Tingkat jaringan adalah maksimum jumlah tautan tetangga dari suatu simpul dalam suatu jaringan. Jika tingkat ini kecil, maka prosedur komunikasi yang melakukan perutean pesan adalah sederhana dan karenanya lebih efisien, karena sejumlah kecil saluran harus dikendalikan oleh prosedur perutean.

Konektivitas tepi adalah ukuran banyaknya jalur antara simpul jaringan. Ini didefinisikan sebagai minimum sejumlah tepi yang harus gagal — untuk dihapus dari jaringan — agar terputus. Semakin besar konektivitas tepi, semakin baik resistensi jaringan terhadap kerusakan. Selain itu, dalam jaringan dengan konektivitas yang lebih tinggi terdapat lebih sedikit pertentangan proses untuk sumber daya komunikasi.

## **MESH**

Garis k-dimensi yang simpulnya disusun dalam array  $P [1..n_1, 1..n_2, \dots, 1..n_k]$  di mana?  $K \ i = 1 \ n_i = p$ , termasuk dalam kelas jaringan jarang. A vertex  $P [i_1, i_2, \dots, i_j, \dots, i_k]$  terhubung ke simpul  $P [i_1, i_2, \dots, i_j \pm 1, \dots, i_k]$  untuk  $j = 1, 2, \dots, k$  jika simpul tersebut ada. Dengan demikian, hampir setiap titik terhubung oleh tepi (tautan) dengan  $2k$  simpul lainnya. menggambarkan diagram satu dimensi mesh di mana setiap simpul kecuali untuk dua simpul di ujung memiliki dua tetangga. Melengkapi jaringan dengan koneksi sampul menghasilkan cincin, atau torus satu dimensi Keuntungan dari jerat satu dimensi adalah Gelar kecil sama dengan dan kemudahan penskalaan, sedangkan kelemahan jaringan besar diameter sama dengan  $p - 1$ . Angka

2.8a dan 2.8b menggambarkan jerat dua dimensi ( $k = 2$ ) derajat 4 dan diameter  $O(\sqrt{p})$ . Struktur serupa memiliki mesh tiga dimensi Keuntungan dari jerat dua dan tiga dimensi adalah bahwa mereka memungkinkan pemetaan komputasi yang mudah dilakukan pada struktur reguler. Contohnya termasuk perhitungan aljabar linier yang melibatkan operasi matriks, atau perhitungan dalam grafik komputer, di mana bidang gambar dipartisi menjadi potongan-potongan reguler yang diproses oleh prosesor individu. Partisi serupa digunakan dalam algoritma pemodelan spasial dari berbagai macam fenomena, seperti aliran udara di atmosfer untuk memprediksi cuaca, aliran cairan dalam hidrodinamika, distribusi medan magnet dalam inti transformer, dll.

## CUBE

dapat berisi sejumlah simpul pada setiap dimensi. Membatasi jumlah ini untuk dua hasil acube, orhypercube, topologi. Kubus  $k$ -dimensi terdiri dari  $p = 2^k$  simpul berjumlah  $0, 1, \dots, 2^k - 1$ . Lebih mudah mewakili angka simpul dalam notasi biner,  $w \in \{0, 1\}^k$ . Dua simpul dihubungkan oleh tepi, jika string sedikit  $w$  sesuai dengan jumlah dari simpul simpul ini berbeda hanya pada satu posisi. Kubus nol dimensi terdiri dari satu simpul tunggal ( $p = 2^0 = 1$ )