Anggota Kelompok:

1. Moody
2. Jose
3. Yosia

Menghitung jumlah array resiprokal dari sebuah vektor atau array melibatkan penambahan kebalikan dari semua elemen array.

Susunan program akan berisi 2 fungsi utama:

1. Random array yang sebesar input yang dimasukkan , boleh menggunakan numpy, dan dilakukan secara sequential.
2. Sum reciprocal yang jumlah element parallelization sebesar input yang dimasukkan (tidak boleh menggunakan numpy)

Dan fungsi pendukung seperti:

1. Menerima input dari argument command line.
2. Menghitung dan menampilkan lama waktu pengerjaan masing-masing fungsi.

Kode yang digunakan :

```python
import threading
import random
import time
import sys

my_array = []
sum = 0.0


class multithread(threading.Thread):
    def __init__(self, id, array, indexEnd, npath):
        threading.Thread.__init__(self)
        self.id = id
        self.array = array
        self.npath = npath
        self.indexEnd = indexEnd

    def run(self):
        reciprocal(self.array, self.indexEnd, self.npath)


def reciprocal(array, indexEnd, npath):
    global sum
    for i in range(int(indexEnd - npath), int(indexEnd)):
        sum = sum + (1 / my_array[i])
```

```python
def make_array(numberArray):
    print("Menggenerate array " + str(numberArray) + " processing dan reciprocal.")
    for x in range(numberArray):
        my_array.append(float(random.randint(1, 99)))


def start_time():
    return time.time()


def end_time(start):
    end = time.time()
    return (end - start)
```

```
def main():

    npath = 0
    numberOfArray = 0
    multi_thread = []
    index_End = 0

    for i, arg in enumerate(sys.argv):
        if (arg == "--thread"):
            npath = int(sys.argv[i + 1])
        if (arg == "--array"):
            numberOfArray = int(sys.argv[i + 1])

    startMakeArray = start_time()
    make_array(numberOfArray)
    print("%s %0.2f %s" % ("Waktu pengerjaan  :", end_time(startMakeArray), "detik\n"))

    startReciprocal = start_time()
    print("Menghitung sum dari array reciprocal processing.")
```

```
    for x in range(npath):
        index_End = index_End + (numberOfArray / npath)
        multi_thread.append(multithread(x, my_array, index_End, numberOfArray / npath))

    for x in range(npath):
        multi_thread[x].start()

    for x in range(npath):
        multi_thread[x].join()
    print("%s %0.2f %s" % ("Waktu pengerjaan  :", end_time(startReciprocal), "detik\n"))

if __name__ == "__main__":
    main()
```

Evaluasi project:

1. **Evaluasi lama program tersebut menggunakan 1 eksekusi dengan random array sebesar 1.000.000 element**

   Berikut adalah input dan output yang kita dapatkan:

```
python arrayreciprocal.py --thread 1 --array 1000000
```

```
Menggenerate array 1000000 processing dan reciprocal.
Waktu pengerjaan  : 0.83 detik

Menghitung sum dari array reciprocal processing.
Waktu pengerjaan  : 0.11 detik
```

2. **Evaluasi lama program tersebut menggunakan 1 eksekusi dengan random array sebesar 100.000.000 element**

```
python arrayreciprocal.py --thread 1 --array 100000000
```

```
Menggenerate array 100000000 processing dan reciprocal.
Waktu pengerjaan  : 87.07 detik

Menghitung sum dari array reciprocal processing.
Waktu pengerjaan  : 251.10 detik
```

3. **Evaluasi lama program tersebut menggunakan N jalur eksekusi dengan random array sebesar 1.000.000 element, sebagai contoh N yang digunakan adalah N = 10**

```
py Thread.py --thread 10 --array 1000000
```

```
Menggenerate array 1000000 processing dan reciprocal.
Waktu pengerjaan  : 1.51 detik

menghitung sum dari array reciprocal processing.
Waktu pengerjaan  : 0.22 detik
```

4. **Evaluasi lama program tersebut menggunakan N jalur eksekusi dengan random array sebesar 100.000.000 element, sebagai contoh N yang digunakan adalah N = 10**

```
py Thread.py --thread 10 --array 100000000
```

```
Menggenerate array 100000000 processing dan reciprocal.
Waktu pengerjaan  : 166.85 detik

menghitung sum dari array reciprocal processing.
Waktu pengerjaan  : 435.83 detik
```