```
from multiprocessing import *
import random
import hashlib
import pyfiglet
from queue import Queue
import time


# Variables Declaration
solNb = 10
nextRound = 1
lastRound = 10
target=100000000
sols = 0
roundID = 0
nonce = 1

def screen():
    result = pyfiglet.figlet_format("TP Course olympique ..!")
    print(result)

screen()

x=random.randint(1, 10000)

q = Queue()

def hash_string(string):
    """
    Return a SHA-256 hash of the given string
    """
    return hashlib.sha256(string.encode('utf-8')).hexdigest()
```

```
Receive=[x, roundID, nonce]
"""
def listenToNewSol():
    name = current_process().name
    print name, 'The time required for implementation and
initiation:'
    if
(hash_string(str(Receive[0])+str(Receive[1])+str(Receive[2]))
< (hash_string(str(target)))):
        global sols
        sols = sols + 1
        q.put(nonce)
        print name, 'I am the winner',hash_string(str(x))
"""
wait = random.uniform(2.5, 10.0)
def listenToNewSo():
    start = time.time()
    name = current_process().name
    time.sleep(wait)
    if
(hash_string(str(Receive[0])+str(Receive[1])+str(Receive[2]))
< (hash_string(str(target)))):
        global sols
        sols =  sols + 1
        q.put(nonce)
        print name, 'I am the winner'
        exit
    end = time.time()
    print "Execution time is:", end-start
if __name__ == "__main__":

# The number of process we want to create here equals =
30
```

```
    number_process = 30
    processes = [Process(target=listenToNewSo, args=()) for
x in range(number_process)]
    if (nextRound <= lastRound):
        if
(hash_string(str(Receive[0])+str(Receive[1])+str(Receive[2]))
> (hash_string(str(target)))):
            nonce = nonce + 1
            exit
        sols = sols + 1
    if (nextRound == lastRound):
          print ('I am Winer')
          exit
    Receive=[x, roundID, nonce]
    q.put(nonce)
    while (sols < solNb):

          if
(hash_string(str(Receive[0])+str(Receive[1])+str(Receive[2]))
< (hash_string(str(target)))):
              sols = sols + 1
              q.put(Receive[2])
              exit
    for p in processes:
        p.start()

    for p in processes:
        p.join()
# Olympic racing games between a set E of process
```