Philadelphia University Faculty of Engineering Department of Computer Engineering		Date:- 10/05/2017 Allowed time:- 60 minutes

Operating Systems (630422) Second Exam

Student Name: - ……..... ID: - ……

Question 1: chose the correct answer for the following: 5 points

1- ____ scheduling is approximated by predicting the next CPU burst with an exponential average of the measured lengths of previous CPU bursts.

A) Multilevel queue B) RR C) FCFS D) SJF

2- With _____ a thread executes on a processor until a long-latency event (i.e. a memory stall) occurs.

A) coarse-grained multithreading B) fine-grained multithreading
C) virtualization D) multicore processors

3- program that has 40% serial portion, then the maximum speed up when using parallel programming.

A) 2 B) 2.5 C) 3 D) 3.5

4- One necessary condition for deadlock is _____, which states that a process must be holding one resource and waiting to acquire additional resources.

A) hold and wait B) mutual exclusion C) circular wait D) no preemption

5. In a system resource-allocation graph, ____.

A) a directed edge from a process to a resource is called an assignment edge
B) a directed edge from a resource to a process is called a request edge
C) a directed edge from a process to a resource is called a request edge
D) None of the above

Question2: What effect does the size of the time quantum have on the performance of an RR algorithm? **2 points**

Question3: what is the main disadvantage of SJF scheduling algorithm and how it can be solved? **2 points**

Question4: given the following information regarding processes (p1,p2,p3,p4) what is the average waiting time for boot FCFS and shortest remaining time first scheduling algorithms. **4 points**

Process	Arrival Time	Burst Time
P_1	0	4
P_2	2	9
P_3	5	6
P_4	6	4

Question5: What are the three general ways that a deadlock can be handled? **3 points**

1-

2-

3-

Question6: trace the following program. What is the number of threads created and what is the output of each thread and main. **4 point**

```
#include <pthread.h>
#include <iostream.h>
#include <stdlib.h>
int NUM_THREADS=4
int sum[NUM_THREADS]={0};

void *calc(void *num)
{
   int n= *(int*)num;
   for(int i=0;i<=n;i++)
        sum[n]+=i;
   cout<<sum[n]<<endl;
   pthread_exit(NULL);
}
int main()
{
   pthread_t threads[NUM_THREADS];
   int rc;
   for(int t=0;t<NUM_THREADS;t++)
   {
      rc = pthread_create(&threads[t], NULL, calc, (void *)t);
      if (rc)
      {
        cout<<"ERROR; return code from pthread_create() is”<<rc<<endl;
        exit(-1);
      }
   }
   for(int i=0;i<NUM_THREADS;i++)
        pthread_join(&threads[i],NULL);


   for(int i=0;i<NUM_THREADS;i++)
        cout<<sum[i]<<”\t”;
    return 0;
}
```