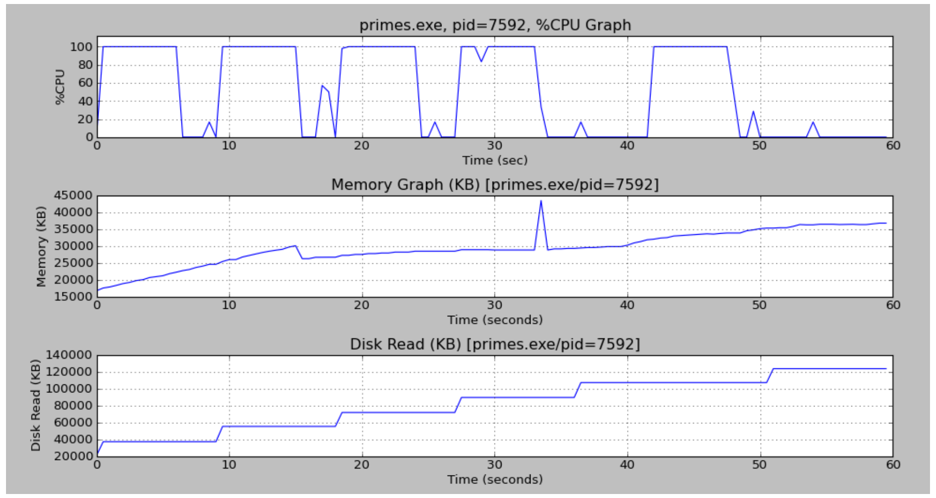
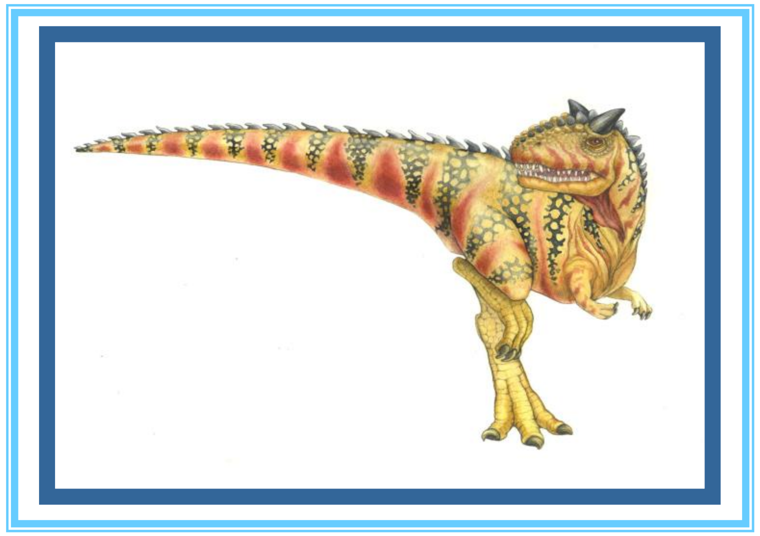
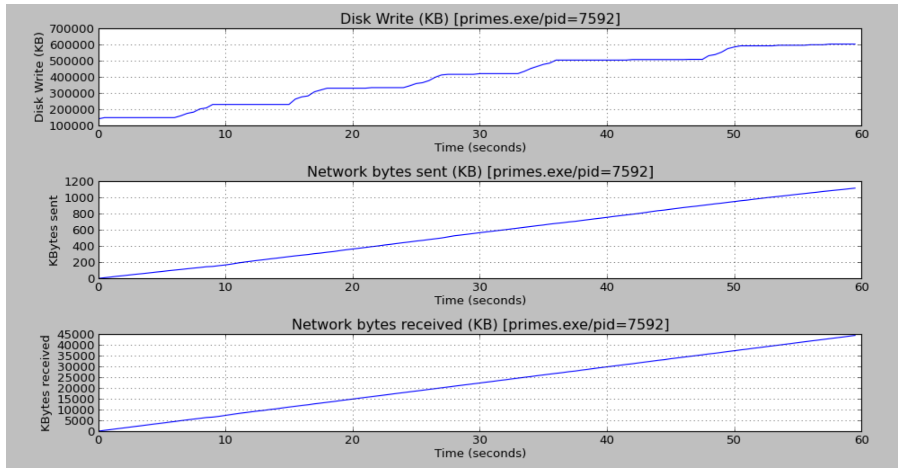
Operating Systems 31261
Course Project
Windows Process Monitor







Files Organization

- Download **os_proj.zip** from: http://www.samyzaf.com/braude/OS/os_proj.zip
- Unzip this file in drive C (or D), so your project will reside in: C:\os_proj (or D:\os_proj)
- You will find there all the files you need for the project
- Make sure to edit the **README.txt** file and enter all the required information (name, email, phones, partner, etc.)
- After completing your project, you should zip this directory back to **os_proj.zip** and upload it to: http://www.samyzaf.com/braude/OS/upload.html

Submission Policy

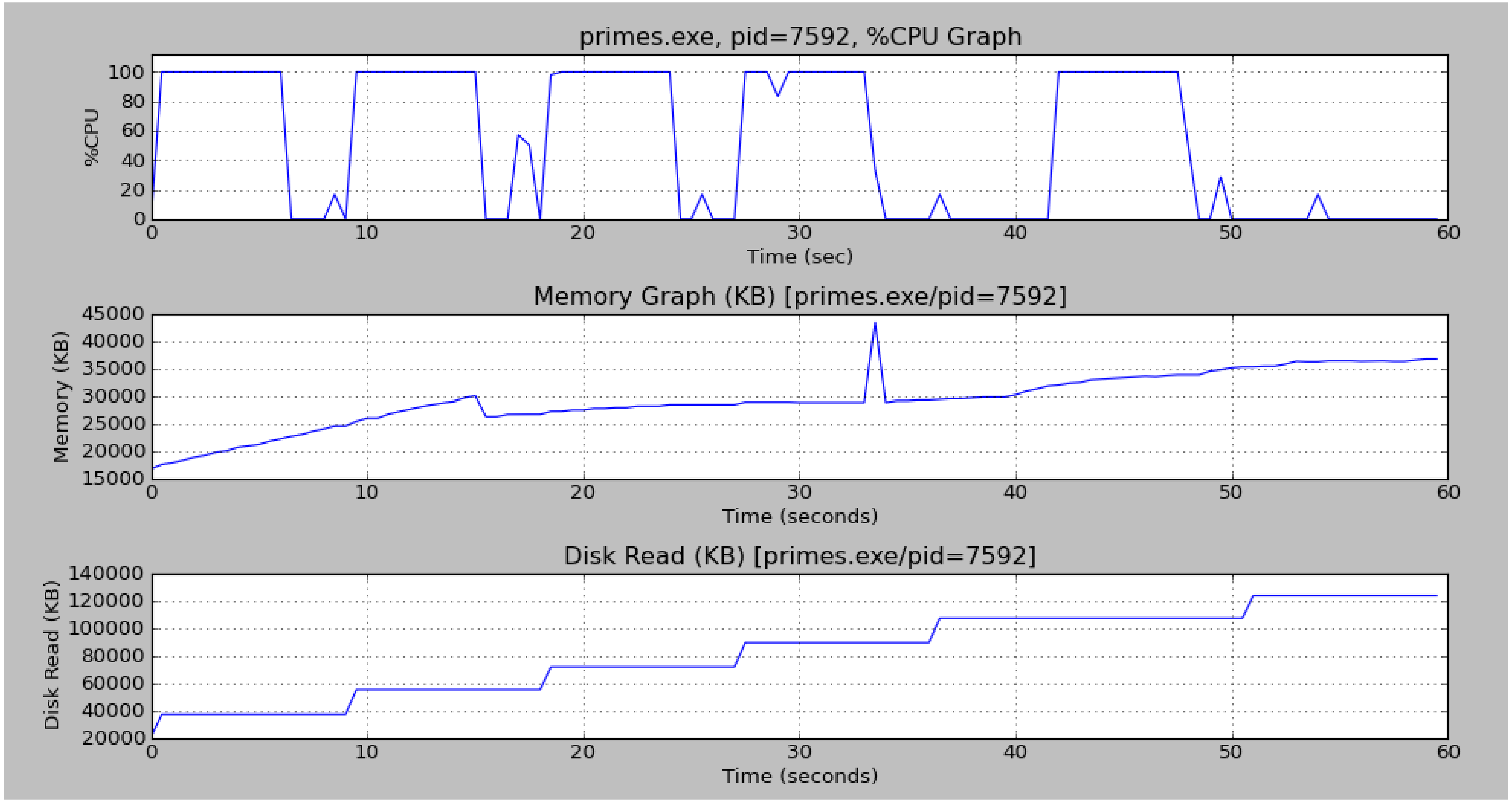
- **Deadline:** June 07, 2014 (till midnight)
  - Upload site will be closed after this date!
- Work in pairs is OK (but not triples!!!)
- A 30 minutes **project review** will be held for each partner separately!
- Make sure to reserve a review slot as early as possible
- Use the scheduling tool to schedule a meeting: http://www.samyzaf.com/cgi-bin/appsched.cgi
- If something is not clear, wrong, or missing, please let me know soon!

samy@samyzaf.com

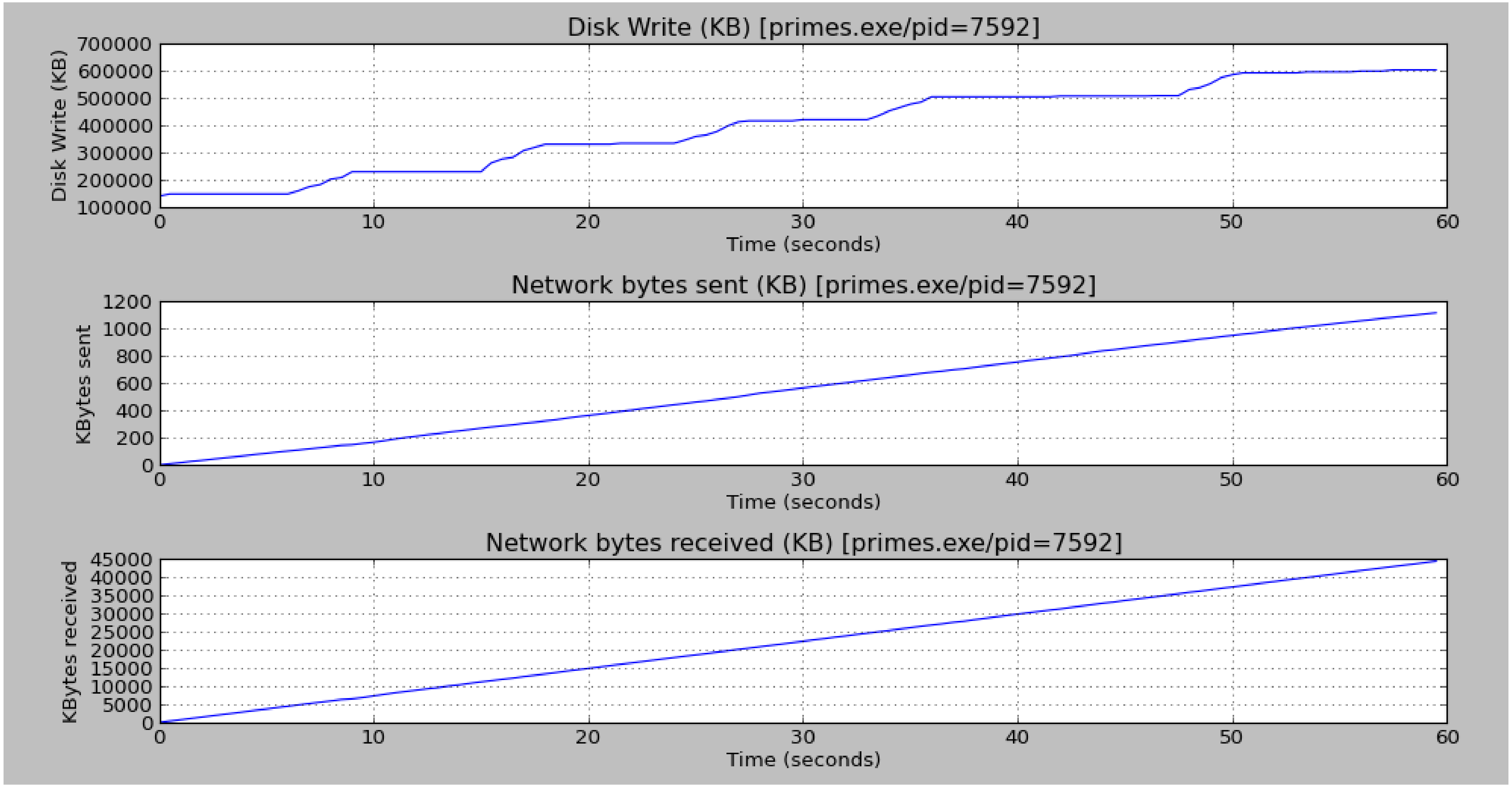
Objectives

- Your mission is to monitor the CPU, memory, disk, and network activities of a **single process** in Windows
- You will use the **primes.exe** program as a test case for testing your work (it is included in the **os_proj.zip** file)
- You need to run the **primes.exe** program monitor its process
- You will have to create **activity graphs** and an **activity report** (as Excel sheet in CSV format)
- More details are explained in the following slides

Your have to create graphs like:

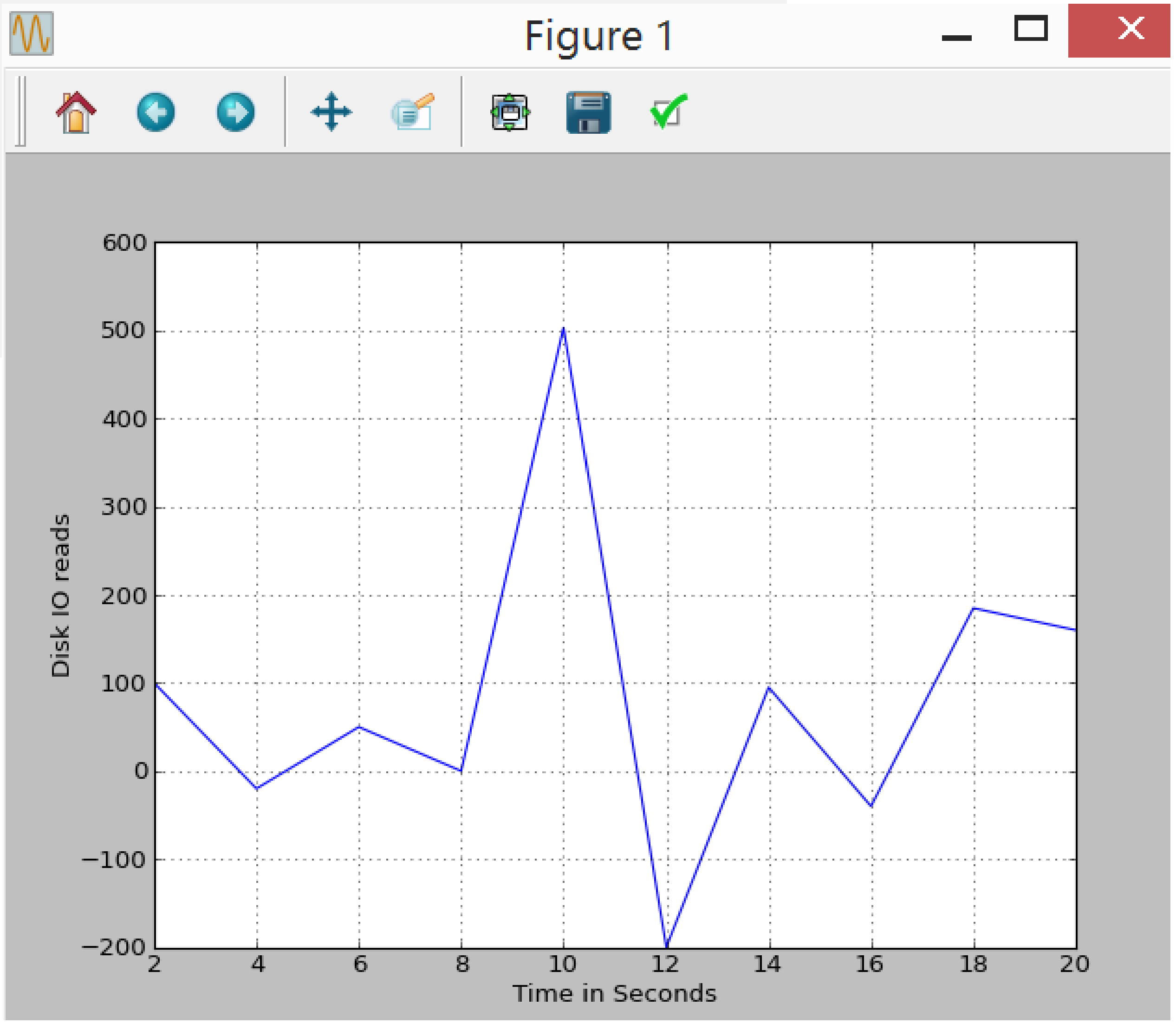


Disk write, Network send/receive graphs



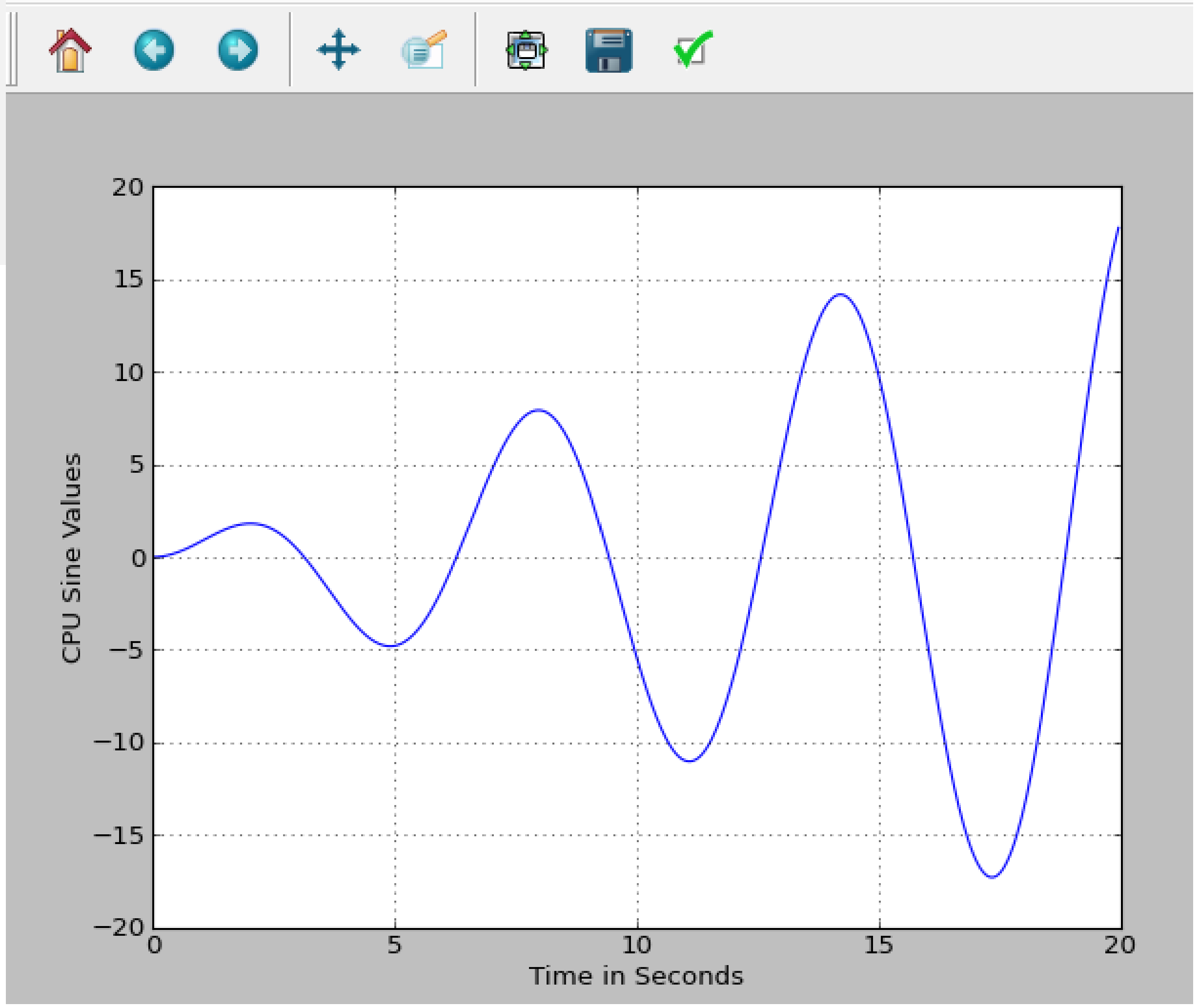
Python Graph Plotting is Simple!

```
import math
import matplotlib.pyplot as mpl

def plot1():
    xvalues = [2, 4, 6, 8, 10, 12, 14, 16, 18, 20]
    yvalues = [100, -20, 50, 0, 503, -200, 95, -40, 185, 160]
    mpl.plot(xvalues, yvalues)
    mpl.xlabel('Time in Seconds')
    mpl.ylabel('Disk IO reads')
    mpl.show()
```



Python Graph Plotting is Simple!

```
import math
import matplotlib.pyplot as mpl

def plot2():
    xvalues = [n * 0.05 for n in range(400)]
    yvalues = [x * math.sin(x) for x in xvalues ]
    mpl.plot(xvalues, yvalues)
    mpl.xlabel('Time in Seconds')
    mpl.ylabel('CPU Sine Values')
    mpl.show()
```



The psutil module

- In this project you will have to do Google search and study! This is the only way to do productive work these days …
- The project is fully based on the Python **psutil** module (this module is already included in Anaconda)
- The **psutil** module is great for finding information on processes and also for manipulating processes
- You will find plenty of information about the **psutil** module in the following links:
  https://code.google.com/p/psutil
  https://code.google.com/p/psutil/wiki/Documentation
- Please start reading and experimenting soon …

The primes.exe program

- The **primes.exe** program is included in os_proj.zip
- You should use it as a test case for this project
- It does the following things
- Using the CPU for computing prime numbers
- Using the disk for read and write activities
- Using the network for sending and receiving data
- It is recommended to put **primes.exe** on your Windows **Desktop** so it will be easy for you to start it for your tests
- *(If you want, you can view and modify the primes.py program – it is also included in the os_proj directory)*

Function: find_process(name)

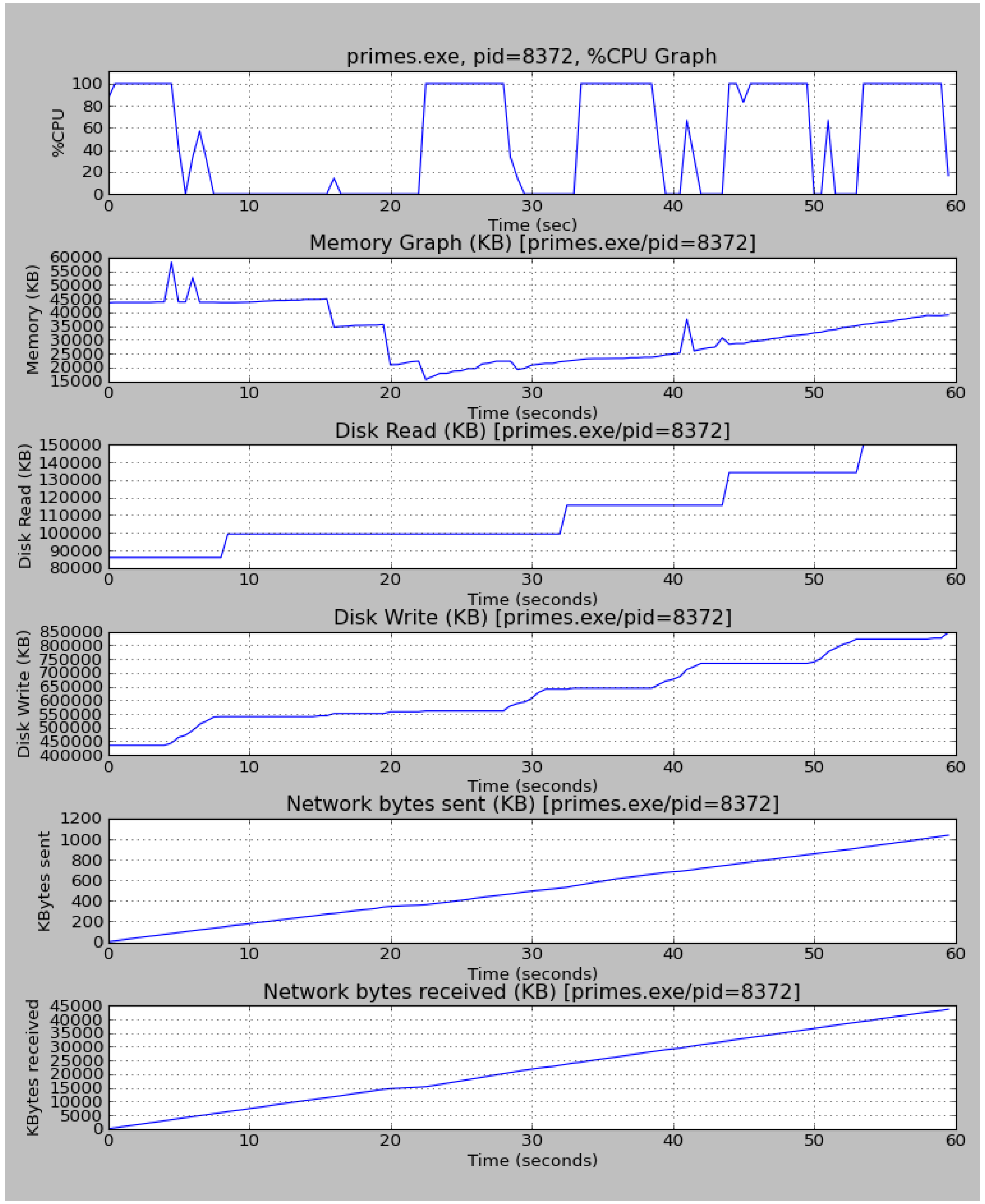
```
# Make sure you understand what is a psutil process object!
# A process object p has many different attributes
# such as p.pid and p.name
# Find process object p such that p.name == name
# Assume you have only one process with this name (otherwise we
# just pick the first one)

def find_process(name):
    # You need to find a process named ‘name’
    # and return its psutil object
    # Otherwise return None
```

process_activity_data(p, n, dt)

```
# Process p activity data is obtained by sampling
# the process p every dt seconds - n times.
# In every sampling we get a list of n values of the following types:
#      cpu_values        list of n cpu percentage values
#      mem_values        list of n memory values (in KB)
#      read_values       list of n disk read sizes (KB)
#      write_values      list of n disk write sizes (KB)
#      net_kbytes_sent   list of n network kbytes sent (KB)
#      net_kbytes_recv   list of n network kbytes received (KB)
# The process_activity_data function should return all
# these lists per process p

def process_activity_data(p, n, dt):
    # Define your function here ...
```

show_activity_graphs(p)

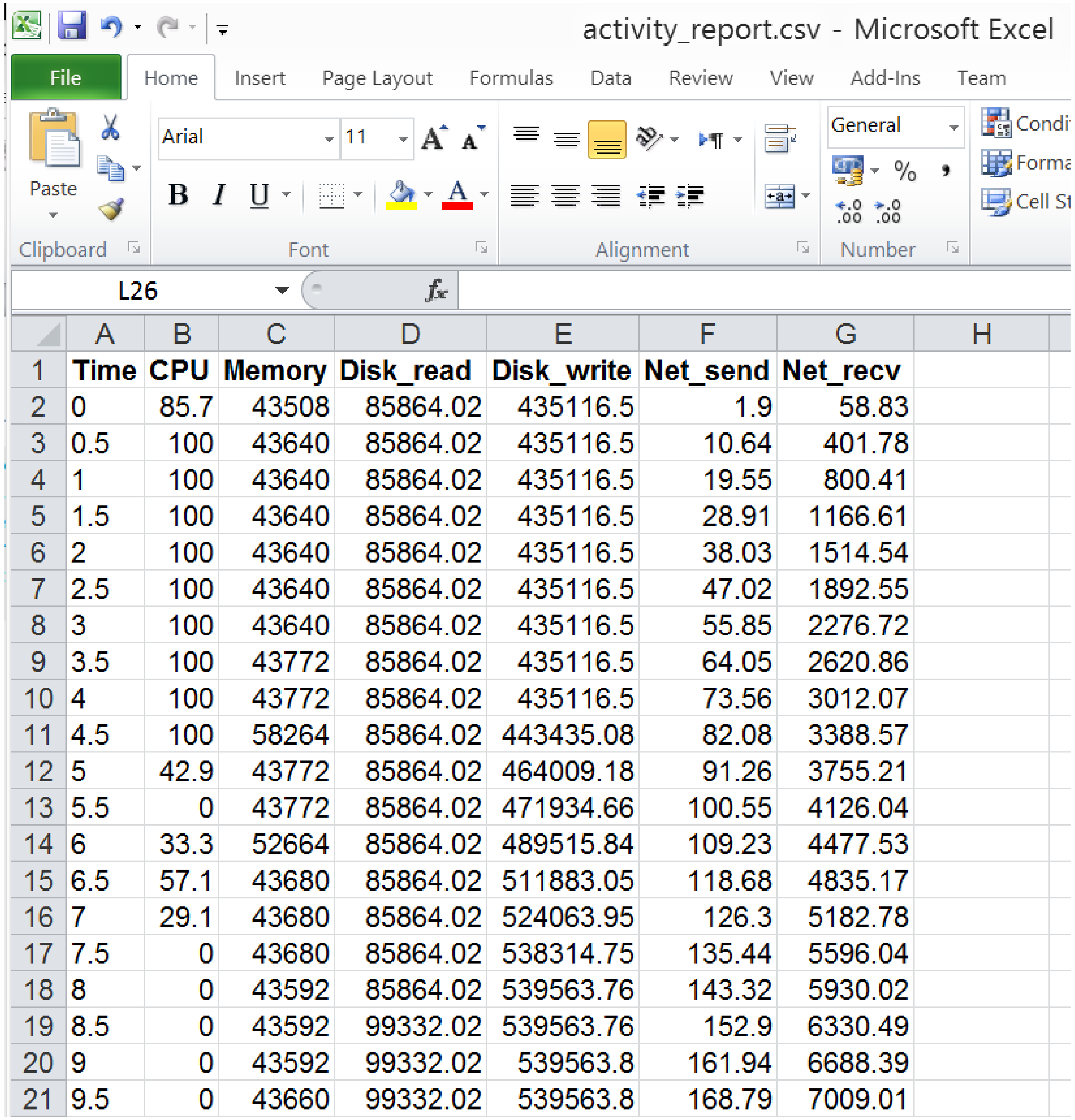
```
# You will need to create 6 different
# graphs in this function!
# Make sure to plot the 6 graphs on
# one page like
# Read about the matplotlib module in:
# http://matplotlib.org
# You don’t have to download it since
# it is included in Anaconda

def show_activity_graphs(p):
    # Define your function here ...
```



EXCEL REPORT

In addition to graphs, you must also create a CSV process activity report. The file name should be ‘activity_report.csv’ and after opening it with Excel it should look like this:

```
def activity_report(p, file):
    # You have to create a CSV
    # text file that contains
    # All process data
```



	A	B	C	D	E	F	G	H
1	Time	CPU	Memory	Disk_read	Disk_write	Net_send	Net_recv	
2	0	85.7	43508	85864.02	435116.5	1.9	58.83	
3	0.5	100	43640	85864.02	435116.5	10.64	401.78	
4	1	100	43640	85864.02	435116.5	19.55	800.41	
5	1.5	100	43640	85864.02	435116.5	28.91	1166.61	
6	2	100	43640	85864.02	435116.5	38.03	1514.54	
7	2.5	100	43640	85864.02	435116.5	47.02	1892.55	
8	3	100	43640	85864.02	435116.5	55.85	2276.72	
9	3.5	100	43772	85864.02	435116.5	64.05	2620.86	
10	4	100	43772	85864.02	435116.5	73.56	3012.07	
11	4.5	100	58264	85864.02	443435.08	82.08	3388.57	
12	5	42.9	43772	85864.02	464009.18	91.26	3755.21	
13	5.5	0	43772	85864.02	471934.66	100.55	4126.04	
14	6	33.3	52664	85864.02	489515.84	109.23	4477.53	
15	6.5	57.1	43680	85864.02	511883.05	118.68	4835.17	
16	7	29.1	43680	85864.02	524063.95	126.3	5182.78	
17	7.5	0	43680	85864.02	538314.75	135.44	5596.04	
18	8	0	43592	85864.02	539563.76	143.32	5930.02	
19	8.5	0	43592	99332.02	539563.76	152.9	6330.49	
20	9	0	43592	99332.02	539563.8	161.94	6688.39	
21	9.5	0	43660	99332.02	539563.8	168.79	7009.01	