Ain Shams University
Faculty of Engineering

Course Assessment Specification (CAS)

Programme Title : **Computer Engineering and Software Systems**
Coursework Title : **Programming Project**
Module Name (UEL) : **Computer Networks**
Course Name (ASU) : **Computer Networks**
Module/Course Code : **EG7533 / CSE351**
Level UEL/ASU : **5 / 3**
UEL Credit Rating : **15 Credits** **ASU Credit Rating** : **3 Credits**
Weighting : **35%**

Maximum mark available:

- 35 on software projects (1 project)

Lecturer : Prof. Ayman M. Bahaa-Eldin

Contact : If you have any issues with this coursework you may contact your lecturer.

Contact details are: Email: ayman.bahaa@eng.asu.edu.eg

Hand-out Date : As shown in submission matrix
Hand-in Date : As shown in submission matrix
Hand-in Method : Submission through LMS
Feedback Date : Your work will be marked and returned within two weeks.

Introduction

This coursework is itemized into several parts to get the 60 marks associated to it.
You must use the templates provided by the instructor to prepare your work.
All assignments and projects will be handed-in electronically, while quizzes and exams are written
.

Learning Outcome to be assessed

2. Explain the use of socket programming in networking applications
4. Recognize the differences between computational and communication overheads
5. Evaluate network performance
10. Work and communicate effectively in a team

Detail of the task

Attached separate file for each task.

89% and above:

Your work must be of outstanding quality and fully meet the requirements of the coursework specification and learning outcomes stated. You must show independent thinking and apply this to your work showing originality and consideration of key issues. There must be evidence of wider reading on the subject. In addition, your proposed solution should:

- illustrate a professional ability of drafting construction details,
- express a deep understanding of the in-hand problem definition,
- and applying, masterly, the learned knowledge in the proposed solution.

76% - 89%:

Your work must be of good quality and meet the requirements of the coursework specification and learning outcomes stated. You must demonstrate some originality in your work and show this by applying new learning to the key issues of the coursework. There must be evidence of wider reading on the subject. In addition, your proposed solution should:

- illustrate a Good ability of drafting construction details,
- express a very Good understanding of the in-hand problem definition,
- and applying most of the learned knowledge, correctly, in the proposed solution.

67% - 76%:

Your work must be comprehensive and meet all of the requirements stated by the coursework specification and learning outcomes. You must show a good understanding of the key concepts and be able to apply them to solve the problem set by the coursework. There must be enough depth to your work to provide evidence of wider reading. In addition, your proposed solution should:

- illustrate a moderate ability of drafting construction details,
- express a good understanding of the in-hand problem definition,
- and applying most of the learned knowledge, correctly, in the proposed solution.

60% - 67%:

Your work must be of a standard that meets the requirements stated by the coursework specification and learning outcomes. You must show a reasonable level of understanding of the key concepts and principles and you must have applied this knowledge to the coursework problem. There should be some evidence of wider reading. In addition, your proposed solution should:

- illustrate a fair ability of drafting construction details,
- express a fair understanding of the in-hand problem definition,
- and applying some of the learned knowledge, correctly, in the proposed solution.

Below 60%:

Your work is of poor quality and does not meet the requirements stated by the coursework specification and learning outcomes. There is a lack of understanding of key concepts and knowledge and no evidence of wider reading. In addition, your proposed solution would be:

- Illustrate an inability of drafting construction details,
- Failed to define the parameters, limitations, and offerings of the in-hand problem,
- Failed to apply correctly the learned knowledge for proposing a valid solution.

Academic Misconduct

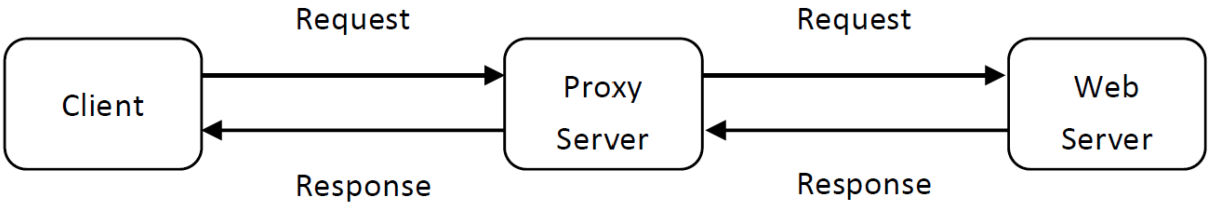
The University defines Academic Misconduct as 'any case of deliberate, premeditated cheating, collusion, plagiarism or falsification of information, in an attempt to deceive and gain an unfair advantage in assessment'. This includes attempting to gain marks as part of a team without making a contribution. The department takes Academic Misconduct very seriously and any suspected cases will be investigated through the University's standard policy. If you are found guilty, you may be expelled from the University with no award.

It is your responsibility to ensure that you understand what constitutes Academic Misconduct and to ensure that you do not break the rules. If you are unclear about what is required, please ask.

HTTP Web Proxy Server with a blacklist URL filter

In this project, you will learn how web proxy servers work and one of their basic functionalities caching. Your task is to develop a small web proxy server which can cache web pages. It is a very simple proxy server which only understands simple GET-requests, but is able to handle all kinds of objects - not just HTML pages, but also images.

Generally, when the client makes a request, the request is sent to the web server. The web server then processes the request and sends back a response message to the requesting client. To improve the performance, we create a proxy server between the client and the web server. Now, both the request message sent by the client and the response message delivered by the web server pass through the proxy server. In other words, the client requests the objects via the proxy server. The proxy server will forward the client's request to the web server. The web server will then generate a response message and deliver it to the proxy server, which in turn sends it to the client.



Code

Below you will find the skeleton code for the proxy server. You are to complete the skeleton code. The places where you need to fill in code are marked with **#Fill in start** and **#Fill in end**. Each place may require one or more lines of code.

Running the Proxy Server

Run the proxy server program using your command prompt and then request a web page from your browser. Direct the requests to the proxy server using your IP address and port number. For example http://localhost:8888/www.google.com

Configuring your Browser

You can also directly configure your web browser to use your proxy. This depends on your browser.

In Internet Explorer, you can set the proxy in Tools > Internet Options > Connections tab > LAN Settings. In Netscape (and derived browsers such as Mozilla), you can set the proxy in Tools > Options > Advanced tab > Network tab > Connection Settings. In both cases you need to give the address of the proxy and the port number that you gave when you ran the proxy server. You should be able to run the proxy and the browser on the same computer without any problem. With this approach, to get a web page using the proxy server, you simply provide the URL of the page you want. For example. http://www.google.com

Requirements

You are required to complete the following code to make the proxy server working and then add the following:

1. *Error Handling*: Currently the proxy server does no error handling. This can be a problem especially when the client requests an object which is not available, since the "404 Not found" response usually has no response body and the proxy assumes there is a body and tries to read it.
2. *Caching*: A typical proxy server will cache the web pages each time the client makes a particular request for the first time. The basic functionality of caching works as follows. When the proxy gets a request, it checks if the requested object is cached, and if yes, it returns the object from the cache, without contacting the server. If the object is not cached, the proxy retrieves the object from the server, returns it to the client and caches a copy for future requests. In practice, the proxy server must verify that the cached responses are still valid and that they are the correct responses to the client's requests. You can read more about caching and how it is handled in HTTP in RFC 2068. Add the simple caching functionality described above. You do not need to implement any replacement or validation policies.

Your implementation, however, will need to be able to write responses to the disk (i.e., the cache) and fetch them from the disk when you get a cache hit. For this you need to implement some internal data structure in the proxy to keep track of which objects are cached and where they are on the disk. You can keep this data structure in main memory; there is no need to make it persist across shutdowns.

3. *URL Filter*: You need to implement a filter, where when you receive a request for a specific URL, you need to look up a local data base and if the URL is listed there, then you should return an error page with a message that says "This URL is blocked". Your local database may be a simple text file where each URL is written in a separate line.

What to Hand in

You will hand in a single PDF document containing your design of the system, complete proxy server code and screenshots at the client side verifying that you indeed get the web page via the proxy server.

Skeleton Python Code for the Proxy Server

```
from socket import *
import sys

if len(sys.argv) <= 1:
    print ('Usage : "python ProxyServer.py server_ip"\n[server_ip : It is the IP Address Of Proxy Server')
    sys.exit(2)

# Create a server socket, bind it to a port and start listening
tcpSerSock = socket(AF_INET, SOCK_STREAM)

# Fill in start.

# Fill in end.

while 1:
    # Start receiving data from the client
    print ('\n\nReady to serve...')
    tcpCliSock, addr = tcpSerSock.accept()
    print ('Received a connection from:', addr)
    message = # Fill in start. # Fill in end.
    print (message)

    # Extract the filename from the given message
    print message.split()[1]
    filename = message.split()[1].partition("/")[2]
    print filename
    fileExist = "false"
    filetouse = "/" + filename
    print filetouse
    try:
        # Check wether the file exist in the cache
        f = open(filetouse[1:], "r")
        outputdata = f.readlines()
        fileExist = "true"
        # ProxyServer finds a cache hit and generates a response message
        tcpCliSock.send("HTTP/1.0 200 OK\r\n")
        tcpCliSock.send("Content-Type:text/html\r\n")
        # Fill in start.

        # Fill in end.
        print ('Read from cache')
    # Error handling for file not found in cache
    except IOError:
        if fileExist == "false":
            # Create a socket on the proxyserver
            c = # Fill in start. # Fill in end.
            hostn = filename.replace("www.","",1)
            print hostn
            try:
                # Connect to the socket to port 80
                # Fill in start.
                # Fill in end.
```

```
            # Create a temporary file on this socket and ask port 80 for the file requested by the client
            fileobj = c.makefile('r', 0)
            fileobj.write("GET "+"http://" + filename + " HTTP/1.0\n\n")
            # Read the response into buffer
            # Fill in start.
            # Fill in end.
            # Create a new file in the cache for the requested file.
            # Also send the response in the buffer to client socket and the corresponding file in the cache
            tmpFile = open("./" + filename,"wb")
            # Fill in start.
            # Fill in end.
        except:
            print ("Illegal request")
    else:
        # HTTP response message for file not found
        # Fill in start.
        # Fill in end.
    # Close the client and the server sockets
    tcpCliSock.close()
# Fill in start.
# Fill in end.
```