1.
network(DDN) IP & /13 => 10.129.4.5 & 11111111.11111000.00000000.00000000 => 10.129.0.0
IP(DDN) 172.81.32.0/20 can be any address from 172.81.32.1 to 172.81.47.254
Netmask(CIDR) /20 => First 20 bits are 1 the rest are 0 => 11111111 11111111 11110000 00000000 => 255.255.240.0
Broadcast(DDN) keep first 13 bits, the rest are 1 => 00001010.10000111.11111111.11111111 => 10.135.255.255

2.
a) taken from https://www.cs.cmu.edu/afs/cs/academic/class/15213-f99/www/class/26/tcpclient.c

b)
```
// tcpclient.c - A simple TCP client
// usage: tcpclient <host> <port>
//
//Jason Millette
//ECE331
//04/26/2018
//Exam 2 problem 2
//

#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <unistd.h>
#include <sys/types.h>
#include <sys/socket.h>
#include <netinet/in.h>
#include <netdb.h>

#define BUFSIZE 10240   //Increased BUFSIZE to capture data

//
// error - wrapper for perror
//
void error(char *msg) {
    perror(msg);
    exit(0);
}

int main(int argc, char **argv) {
    int sockfd, portno, n;      //declaring variables and structures
    struct sockaddr_in serveraddr;
    struct hostent *server;
    char *hostname;
    char buf[BUFSIZE] = "GET /index.html HTTP/1.1\r\nhost: myhost\r\n\r\n"; //HTTP request string

    // check command line arguments
    if (argc != 3) {
       fprintf(stderr,"usage: %s <hostname> <port>\n", argv[0]);
       exit(0);
    }
    hostname = argv[1];     //assining user input to appropriate variables
    portno = atoi(argv[2]);

    // socket: create the socket
    sockfd = socket(AF_INET, SOCK_STREAM, 0);
    if (sockfd < 0)
        error("ERROR opening socket");

    // gethostbyname: get the server's DNS entry
    server = gethostbyname(hostname);
    if (server == NULL) {
        fprintf(stderr,"ERROR, no such host as %s\n", hostname);
        exit(0);
    }
```

```
    // build the server's Internet address
    bzero((char *) &serveraddr, sizeof(serveraddr));
    serveraddr.sin_family = AF_INET;
    bcopy((char *)server->h_addr,
          (char *)&serveraddr.sin_addr.s_addr, server->h_length);
    serveraddr.sin_port = htons(portno);

    // connect: create a connection with the server
    if (connect(sockfd, &serveraddr, sizeof(serveraddr)) < 0)
      error("ERROR connecting");
    // Removed the call for message to be sent. Using the pre defined string
    // send the message line to the server
    n = write(sockfd, buf, strlen(buf));
    if (n < 0)
      error("ERROR writing to socket");

    // print the server's reply
    bzero(buf, BUFSIZE);
    n = read(sockfd, buf, BUFSIZE);
    if (n < 0)
      error("ERROR reading from socket");
    printf("Echo from server: %s", buf);
    close(sockfd);
    return 0;
}
```

3.

Host Elessar

Kernel IP routing table

Destination	Gateway	Genmask	Flags	Metric	Ref	Use	Iface
1.2.4.0	0.0.0.0	255.255.128.0	UH	2	4	2	ETH1
10.0.0.0	0.0.0.0	255.254.0.0	U	3	0	0	ETH0
1.2.0.0	0.0.0.0	255.255.0.0	U	1	2	1	ETH1
0.0.0.0	1.2.4.1	0.0.0.0	UG	2	0	0	ETH1

Host Legolas

Kernel IP routing table

Destination	Gateway	Genmask	Flags	Metric	Ref	Use	Iface
141.114.3.0	0.0.0.0	255.255.255.192	UH	3	0	0	ETH0
0.0.0.0	141.114.3.1	0.0.0.0	UG	0	0	0	ETH0

4.

Host MAC	Destination MAC	Host IP	Destination IP
00:00:00:00:00:11	55:00:00:00:00:00	10.1.2.3	10.1.2.10
00:00:00:00:00:11	66:00:00:00:00:00	10.1.2.10	141.114.3.10
00:00:00:00:00:11	00:00:00:00:00:22	141.114.3.10	141.114.3.3

5.

Host MAC	Destination MAC	Host IP	Destination IP
00:00:00:00:00:22	66:00:00:00:00:00	141.114.3.3	141.114.3.10
00:00:00:00:00:22	77:00:00:00:00:00	141.114.3.10	1.2.3.100
00:00:00:00:00:22	FF:00:00:00:00:00	1.2.3.100	1.2.4.100

6.

```
<?php
function Table($width, $height){
    echo "<table border ='2'; cellspacing='0'; >";      #definition of table
    $row=0;             #definition of variables
    $column=0;
    for($row = 0; $row < $height; $row++){              #runs through every row
        echo "<tr>"; #starts row
        for($column = 0; $column < $width; $column++){
           echo "<td align='center'>".($row+$height*$column)."</td>";   #prints the column
        }               #aligns number to center of cell

        echo"</tr>";            #ends ends row
    }
```

```
    echo "</table>";            #ends table
}
Table(5, 4);   #calls Table
?>
```

7.

```
SELECT "date text", "time text", "price real" * FROM bitcoin WHERE "price real" = (SELECT M
AX("price real") AND date('now','-30 day'));
```

8.

```
#!/usr/bin/perl
    #open the file with the given dictionary
    open(my $file,"<","/usr/share/dict/american-english-large") or die "$!";

    while (defined($letter = getc($file))) {        #gets the quantity of each letter and f
ills hash
        $frequency{$letter}++ if $letter =~/[[:alpha:]]/;
    }

    while (my($letter, $count) = each %frequency) {     #finds max count
        if ($max < $count) {
            $max = $count;
        }
    }

    #prints the letter then the correct number of * normalized to 70
    print $_, "*" x (($frequency{$_} / $max) * 69), "\n" for sort keys %frequency;

    close $file;
```

9.

There is a definite security risk to leaving this process running. You should not stop this process as you do not have permission to do so. Your administrative access was given for a specific purpose and not security. The correct thing to do would be to immediately contact someone who is in charge of security, and show them what you found.