4-hour Written Re-Exam in Computer Systems

Department of Computer Science, University of Copenhagen (DIKU)
Date: April 17, 2018

Preamble

Solution

Disclaimer: The reference solutions in the following range from exact results to sketch solutions; this is entirely on purpose. Some of the assignments are very open, which reflects to our solutions being just one of many. Of course we try to give a good solution and even solutions that are much more detailed than what we expect you to give. On the other hand, you would expect to give more detailed solutions that our sketches. Given the broad spectrum of solutions, it is not possible to directly infer any grade level from this document. All solutions have been written in the in-lined space with this colour.

This is the exam set for the 4 hour written re-exam in Computer Systems (CompSys), B1+2-2017/18. This document consists of 20 pages excluding this preamble; make sure you have them all. Read the rest of this preamble carefully. Your submission will be graded as a whole, on the 7-point grading scale, with external censorship.

- You can answer in either Danish or English.
- Remember to write your exam number on all pages.
- You do not have to hand-in this preamble.

Expected usage of time and space

The set is divided into sub-parts that each are given a rough guiding estimate of the time needed. However, your exact usage of time can differ depending on prior knowledge and skill.

Furthermore, all questions includes formatted space (lines, figures, tables, etc.) for in-line answers. Please use these as much as possible. The available spaces are intended to be large enough to include a satisfactory answer of the question; thus, full answers of the question does not necessarily use all available space.
If you find yourself in a position where you need more space or have to redo (partly) an answer to a question, continue on the backside of a paper or write on a separate sheet of paper. Ensure that the question number is included and that you in the in-lined answer space refers to it; e.g. write "*The [rest of this] answer is written on backside of/in appended page XX.*"

For the true/false and multiple-choice questions with one right answer give only one clearly marked answer. If more answers are given, it will be interpreted as incorrectly answered. Thus, if you change your answer, make sure that this shows clearly.

Exam Policy

This is an *individual*, open-book exam. You may use the course book, notes and any documents printed or stored on your computer, but you may not search the Internet or communicate with others to answer the exam.

Exam number: _____

Errors and Ambiguities

In the event of errors or ambiguities in the exam text, you are expected to state your assumptions as to the intended meaning in your answer. Some ambiguities may be intentional.

IMPORTANT
It is important to consider the context and expectations of the exam sets. Each question is designed to cover one of more learning goals from the course. The total exam set will cover all or (more likely) a subset of all the learning goals; this will vary from year to year.
The course has many more learning goals than are realistic to cover during a 4-hour exam. And even though we limit the tested learning goals, it will still be too much.
Therefore, it is not expected that you give full and correct answer to all questions. Instead you can focus on parts in which you can show what you have learned.
It is, however, important to note that your effort will be graded as a whole. Thus, showing your knowledge in a wide range of topics is better that specialising in a specific topic. This is specially true when considering the three overall topics: Arc, OS, and CN.

Specifically, for this exam set it was need to solve:
* about 40 % to pass
* about 55 % to get a mid-range grade
* about 70 % to get a top grade
* no one solved all parts correctly!

NB! we adjust the exam set from year to year, so the above is not written in stone and can change slightly for your exam.

4-hour Written Re-Exam in Computer Systems
Department of Computer Science, University of Copenhagen
Date: April 17, 2018

Exam number: _____

1 Machine architecture (about 33 %)

1.1 True/False Questions (about 3 %)

For each statement, answer True or False. (Put one "X" in each.)	True	False
a) Within Boolean arithmetic then `~(A & B) = (~A)^(~B)`.		X
b) The largest `unsigned char` has the value 256.		X
c) The lowest `signed char` has the value -128.	X	
d) Assume `x` and `y` are signed natural values (e.g. `long`), then the C expression `(x < y) == ((-x) > (-y))` is always evaluated to true.		X
e) In the Linux call model, the return address of a procedure call is located in a special purpose register.		X

a) is false, though is looks like DeMorgan's law. Write truth table or consider the size of the domains.
d) is false as the arithmetic negation of the smallest (negative) value of a two's complement number does not have positive representative. Thus for this number then `x == -x`.
e) is false as the return address is located on the stack.

1.2 Multiple Choice Questions (about 3 %)

In each of the following questions choose one answer.

Multiple Choice Questions, 1.2.1: In a pipelined architecture, resolving correctness of a predicted jump is performed in the:

☐ **a)** Fetch phase (`F`),

☐ **b)** decode phase (`D`),

☒ **c)** execute phase (`X`), or

☐ **d)** memory phase (`M`).

Multiple Choice Questions, 1.2.2: The 6-bit two's complement number `100101` represents the value

☐ **a)** 37,

☐ **b)** 27,

☐ **c)** 17,

☐ **d)** -17,

☒ **e)** -27, or

☐ **f)** -37.

Exam number: _____

1.3 Data Cache (about 7 %)

Given a byte-addressed machine with 16-bit addresses. The machine is equipped with a 4-way set associative data cache of 8 kilobytes. Cache have a block size of 16 bytes.

Data Cache, 1.3.1: For each bit in the table below, indicate which bits of the address would be used for

- block offset (denote it with `O`),
- cache tag (denote it with `T`), and
- set index (denote it with `S`).

T	T	T	T	T	S	S	S	S	S	S	S	O	O	O	O
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

Data Cache, 1.3.2: Consider the stream of storage references in hexadecimal format:

`0xA010` , `0xF020` , `0xFF20` , `0xFF0C` , `0x0028` , `0xF0A4` , `0xF034` .

Assuming that the cache initially is cold followed by referencing the above stream. Given that the cache uses LRU replacement, what is the effect of a following reference to address: `0x0024`.

Block Offset:	`0x4`
Set Index:	`0x02`
Cache tag:	`0x00`
Cache hit or miss:	hit
In case of cache miss, which cache tag is evicted:	

Data Cache, 1.3.3: Briefly argument for your answer regarding cache hit/miss and address eviction and show the content of the set before and after the reference.

The address is already in the cache as the block containing it was loaded with `0x0028`. The cache is 4-way and the only other following block loaded into the same set `0x01` is `0xF034`.

Exam number: _____

1.4 Assembler programming (about 12 %)

Consider the following program written in X86prime-assembler.

The following program have been updated from x86-64 assembler to x86prime.

```
1  program:
2    movq  %rdi, %rax
3    movl  $0, %edx
4    jmp L2
5  L3:
6    addq  (%rax), %rdx
7    addq  $8, %rax
8  L2:
9    cmpq  %rsi, %rdx
10   jl  L3
11   subq  %rdi, %rax
12   sarq  $2, %rax
13   ret
```

Assembler programming, 1.4.1: Rewrite the above X86prime-assembler program to a C program and explain the functionality of the program. The resulting program should not have a goto-style and minor syntactical mistakes are acceptable.

The program take two arguments (`%rdi` and `%rsi`) and calculates twice the number of elements in an unbounded array (*a) that is a required to reach max.

```
long program(long *arr, long max) {
  long *arr_ptr = arr; // Located in %rdi
  long sum = 0; // Located in %rdx
  while (sum < max) {
    sum += *arr_ptr;
    arr_ptr++;
  }
  return (arr_ptr - arr) * 2;
}
```

That the return values was twice the size was unintended and due to a copy-paste error; the instruction should have right-shifted by 3. Any answer that does not include this is also accepted and this part should show understanding of pointer arithmetic.

Exam number: _____

Assembler programming, 1.4.2: Argument for your choice of statements and expression. Specify which part of the program are not directly translated and argue why.

We see on the register types that values are `long`. First argument (`%rdi` which is used in `%rax`) is a pointer to an array; this is seen by the indexing in line 6. The second (`%rsi`) is a number. `%rdx` is overwritten by 0 (in `%edx`) and therefore not an input.

The program structure is a standard while loop as following translation from BOH. The comparison of `%rsi` to `%rdx` with the following jump-less-than is the condition of `sum` < lstinlinemax.

The array is iterated using pointer arithmetic which is seen in line 7. Line 11 finds the number of elements from the initial array pointer, which in line 12 is divided by 4 using a right-shift. (Note, this was intended to be 8 (shift 3) to account for the length a `long`; thus the return value is twice the length.)

Assembler programming, 1.4.3: Briefly describe the semantic difference between logical and arithmetical shifts.

Right shifts are identical. A logical left shift always adds a 0 as the most significant bit. Arithmetical left shifts duplicated the most significant bit to ensure the sign of a two's complement number.

Exam number: _____

1.5 Pipeline (about 8 %)

Assume that we execute on the standard 5-phased pipeline architecture without any kind of forwarding available. Thus it will have the following phases:

- `F` for fetch,
- `D` for decode,
- `X` for execute,
- `M` for memory, and
- `W` for writeback.

Consider the following straight-line code with the straightforward indication of a pipeline-diagram.

```
1  Code                        Timing
2
3  movq  (%rax), %rdx          FDXMW
4  subq  %rdi, %rax             FDXMW
5  sarq  $2, %rax                FDXMW
6  addq  (%rax), %rdx             FDXMW
```

Pipeline, 1.5.1: Identify the types of hazards and location in the above code and provide the correct pipeline-diagram with delay of the correct phases.

Values are written back to registers in W phase and read in D phase.

Line 3 reads a the content of `%rax` and there is thus no hazards in line 4.

Line 4 updates `%rax` so line 5 has a data hazard, which delays in D phase; value must be ready in D after W.

Similarly, line 6 reads `%rax`, which creates a data hazard with line 5. First F is delayed for previous delay of D and then it must be delayed in D.

	Code	Timing															
3	`movq (%rax), %rdx`	F	D	X	M	W											
4	`subq %rdi, %rax`		F	D	X	M	W										
5	`sarq $2, %rax`			F	D	D	D	D	X	M	W						
6	`addq (%rax), %rdx`				F	.	.	.	D	D	D	D	X	M	W		

Exam number: _____

2 Operating Systems (about 80 minutes %)

2.1 True/False Questions (about 8 minutes %)

For each statement, answer True or False. (Put one "X" in each.)	True	False
a) `fopen()` is not a system call.	X	
b) There is never more physical memory than virtual memory.		X
c) System calls are implemented via signals.		X
d) Virtual memory requires a disk.		X
e) System calls run in user mode.		X
f) Condition variables cannot be efficiently implemented solely with mutexes.	X	

2.2 Multiple Choice Questions (about 12 minutes %)

In each of the following questions, you may put one or more answers.

Multiple Choice Questions, 2.2.1: Which of the following operations are guaranteed to execute atomically?

☐ **a)** `pthread_cond_signal()`

☒ **b)** `pthread_mutex_lock()`

☐ **c)** `x++` (when `x` is `int`)

☐ **d)** `memcpy(&x, &y, sizeof(x))`

☒ **e)** `pthread_cond_wait()`

☐ **f)** `exit(0)`

Multiple Choice Questions, 2.2.2: Consider a demand-paged system with the following time-measured utilisations:

CPU utilisation	50%
Paging disk	0.7%
Other I/O devices	75%

Which of the following would likely improve CPU utilisation?

☒ **a)** Install a faster CPU.

☐ **b)** Install a bigger paging disk.

☐ **c)** Install a faster paging disk.

☐ **d)** Install more main memory.

☒ **e)** Increase the degree of multiprogramming.

Exam number: _____

2.3 Short Questions (about 24 minutes %)

Short Questions, 2.3.1: Consider a system with the following properties:

- Memory is byte-addressed.
- Virtual addresses are 13 bits wide.
- Physical addresses are 14 bits wide.
- The page size is 64 bytes.
- The TLB is 3-way set associative with four sets and 12 total entries. Its initial contents are:

Set	Tag	PPN	Valid	Tag	PPN	Valid	Tag	PPN	Valid
0	12	34	1	00	00	0	00	00	1
1	0A	10	0	11	15	1	1F	2F	1
2	11	15	0	0F	10	1	07	12	1
3	13	21	1	00	12	0	10	0A	1

- The page table contains 8 PTEs:

VPN	PPN	Valid	VPN	PPN	Valid	VPN	PPN	Valid	VPN	PPN	Valid
00	00	1	21	10	0	3F	23	1	12	34	1
21	11	1	01	02	1	02	01	1	13	43	1

Note that all addresses are given in hexadecimal. In the following questions, you are asked, for various virtual addresses, to show the translation from virtual to physical addresses in the memory system just described. *Hint: there is one TLB hit, one page table hit, and one page fault (not necessarily in that order). This should help you double-check your work.*

Virtual address: `0x1214`

1. Bits of virtual address

12	11	10	9	8	7	6	5	4	3	2	1	0
1	0	0	1	0	0	0	0	1	0	1	0	0

2. Address translation

Parameter	Value
VPN	48
TLB index	0
TLB tag	12
TLB hit? (Y/N)	Y
Page fault? (Y/N)	N
PPN	34

3. Bits of physical address

13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	1	1	0	1	0	0	0	1	0	1	0	0

Exam number: _____

Virtual address: `0x00f0`

1. Bits of virtual address

12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	0	1	1	1	1	0	0	0	0

2. Address translation

Parameter	Value
VPN	3
TLB index	3
TLB tag	0
TLB hit? (Y/N)	N
Page fault? (Y/N)	Y
PPN	

3. Bits of physical address

13	12	11	10	9	8	7	6	5	4	3	2	1	0

Virtual address: `0x084a`

1. Bits of virtual address

12	11	10	9	8	7	6	5	4	3	2	1	0
0	1	0	0	0	0	1	0	0	1	0	1	0

2. Address translation

Parameter	Value
VPN	21
TLB index	01
TLB tag	08
TLB hit? (Y/N)	N
Page fault? (Y/N)	N
PPN	11

3. Bits of physical address

13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	1	0	0	0	1	0	0	1	0	1	0

Exam number: _____

Short Questions, 2.3.2: This problem tests your understanding of exceptional control flow in C programs. For programs A-C, indicate how many "hello" output lines the program would print.
Caution: Don't overlook the `printf()` function in `main()`.

Program A

```
void doit() {
  fork();
  printf("hello\n");
  fork();
  return;
}

int main() {
  doit();
  printf("hello\n");
  exit(0);
}
```

6

Program B

```
void* doit(void* unused) {
  if (fork() == 0) {
    fork();
    printf("hello\n");
    exit(0);
  }
  return NULL;
}

int main() {
  pthread_t t;
  pthread_create(&t, NULL, doit, NULL);
  fork();
  printf("hello\n");
  pthread_join(&t);
  exit(0);
}
```

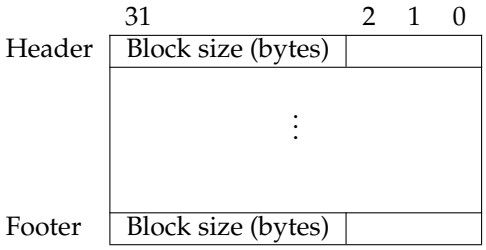
4

Exam number: _____

2.4 Long Questions (about 36 minutes %)

Long Questions, 2.4.1: The following problem concerns dynamic storage allocation.

Consider an allocator that uses an implicit free list. The layout of each allocated and free memory block is as follows, with one 32-bit word per row:



Each memory block, either allocated or free, has a size that is a multiple of eight bytes. Thus, only the 29 higher order bits in the header and footer are needed to record block size, which includes the header and footer. The usage of the remaining 3 lower order bits is as follows:

- `bit 0` indicates the use of the current block: 1 for allocated, 0 for free.
- `bit 1` indicates the use of the previous adjacent block: 1 for allocated, 0 for free.
- `bit 2` is unused and is always set to be 0.

Given the contents of the heap shown on the left, show the new contents of the heap (in the right table) after a call to `free(0x200b010)` is executed. Your answers should be given as hex values. Note that the address grows from bottom up. Assume that the allocator uses immediate coalescing, that is, adjacent free blocks are merged immediately each time a block is freed.

Exam number: _____

Address	Value
0x200b030	0x00000013
0x200b02c	0x200b611c
0x200b028	0x200b512c
0x200b024	0x00000013
0x200b020	0x0000001b
0x200b01c	0x200b511c
0x200b018	0x200b711a
0x200b014	0x200b601c
0x200b010	0x200b601c
0x200b00c	0x0000001b
0x200b008	0x00000012
0x200b004	0x200b601c
0x200b000	0x200b511c
0x200affc	0x00000012

Address	Value
0x200b030	0x00000011
0x200b02c	0x200b611c
0x200b028	0x200b512c
0x200b024	0x00000011
0x200b020	0x0000002a
0x200b01c	0x200b511c
0x200b018	0x200b711a
0x200b014	0x200b601c
0x200b010	0x200b601c
0x200b00c	0x0000001b
0x200b008	0x00000012
0x200b004	0x200b601c
0x200b000	0x200b511c
0x200affc	0x0000002a

Exam number: _____

Long Questions, 2.4.2: Which of the following programming techniques and data structures are "good" for a demand-paged environment, and which are "bad" (performance-wise)? Explain your answers.

- Stack
- Hash table
- Sequential search of array
- Sequential search of linked list
- Binary search of array
- Vector operations (such as vector addition or computing dot products)

A stack is efficient because operations exhibit good *locality*—we are always operating on addresses near each other, which minimises page faults. A hash table can be inefficient, because a good hash algorithm will ensure that accesses are evenly distributed among buckets. A sequential search of an array is efficient, because of good locality. Sequentially searching of a linked list is likely inefficient, because logically neighboring nodes can be arbitrarily distant in memory. Binary search of an array is likewise also inefficient (from a memory access point of view), again because of large jumps in addresses. Vector operations tend to be efficient, because they involve sequentially traversing arrays.

Exam number: _____

3 Computer Networks (about 80 minutes %)

3.1 True/False Questions (about 8 minutes %)

For each statement, answer True or False. (Put one "X" in each.)	True	False
a) Implementation of link layer protocols span both hardware (network controllers) and software (operating systems).	X	
b) Peer-to-peer architectures exhibit better scalability because adding peers results in an increase in cumulative bandwidth available for all communicating parties.	X	
c) For a TCP connection, the receive window can never become zero.		X
d) Convergence time of OSPF protocol is independent of the number of edges in a network.	X	

3.2 Multiple Choice Questions (about 15 minutes %)

In each of the following questions choose one answer.

Multiple Choice Questions, 3.2.1: Consider a two dimensional even parity scheme for error detection. Using this scheme compute the parity bits of the 8-bit ASCII[1] representation of "REEXAM" where each byte of the word forms a row for the two dimensional parity scheme. The resultant parity bits (row followed by column parity bits) are

☒ **a)** 111100 00000110

☐ **b)** 111101 00000010

☐ **c)** 100101 00001110

☐ **d)** None of the above

Multiple Choice Questions, 3.2.2: The broadcast address of the network 117.18.31.54/18 is

☐ **a)** 117.18.31.255

☒ **b)** 117.18.63.255

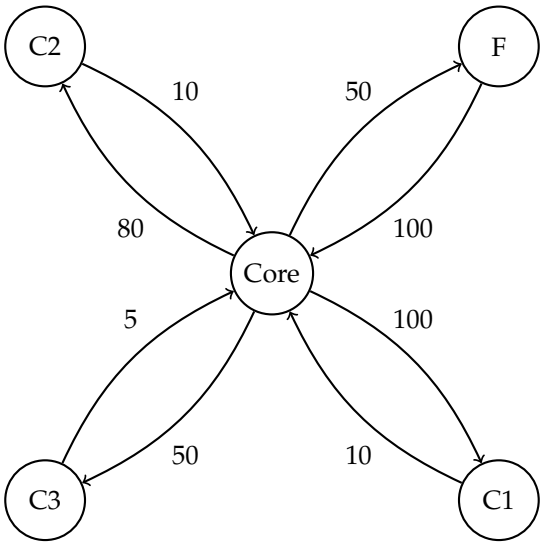
☐ **c)** 117.18.127.255

☐ **d)** None of the above

[1] ASCII codes of A-Z lie contiguously between decimal numbers 65-90.

Exam number: _____

Multiple Choice Questions, 3.2.3: Consider a network as shown in figure below.



The network shows a file server F and clients C1, C2 and C3 connected by a core network X. The directed edges in the network represent the upload and download bandwidth available to the file server and clients from the network core. If the file server and the clients concurrently engage in *peer-to-peer file distribution*, calculate the time to distribute a file of size 10 GB among all the clients. (Assume that the core network bandwidth is not the bottleneck.)

All numbers in the figure are in Mbps. This is not clear from the text, but after a question students were informed by this.

☐ **a)** 8000 sec

☐ **b)** 1920 sec

☐ **c)** 1600 sec

☒ **d)** None of the above

Exam number: _____

3.3 Short Questions (about 24 minutes %)

Short Questions, 3.3.1: Why is an ARP query sent within a broadcast frame while the ARP response is sent within a frame having a specific destination address?

An ARP query is sent to resolve an IP address to a MAC address by the link layer. Since the destination MAC address is unknown, a link layer frame consisting of the ARP query is broadcast using the special MAC broadcast address. Since an ARP response is sent on receipt of an ARP query which contains the MAC address of the sender, the link layer frame containing the ARP response can be sent to the specific MAC address of the ARP query sender.

Short Questions, 3.3.2: Consider the following sequences of HTTP request and response messages between the DIKU web-server (www.diku.dk) and a telnet client.

```
<Request Message 1 follows>
HEAD /~bonii/ HTTP/1.1
Host: www.diku.dk

<Response Message 1 follows>
HTTP/1.1 200 OK
Date: Mon, 26 Mar 2018 06:46:59 GMT
Server: Apache/2.4.6 (Red Hat Enterprise Linux) OpenSSL/1.0.2k-fips mod_fcgid/2.3.9
Last-Modified: Fri, 21 Jul 2017 09:32:54 GMT
ETag: "40b8-554d08cbc59b0"
Accept-Ranges: bytes
Content-Length: 16568
Content-Type: text/html

<Request Message 2 follows>
HEAD /~bonii/ HTTP/1.1
Host: www.diku.dk
If-Modified-Since: Mon, 26 Mar 2018 06:47:01
```

What is the difference between GET and HEAD methods in HTTP? What would be the response from the web server at www.diku.dk on receipt of request message 2 and why?

A HEAD request method is identical to a GET method except that the HTTP response from the server

does not contain a message body.

The response from the web-server on receipt of request message 2 would be 304 Not Modified. This is because request message 2 consists of a conditional GET which only requests the resource if it has been modified since 26 Mar 2018 06:47:01 (the lack of specification of time zone is immaterial here, some telnet clients might interpret it differently). Since the response message 1 shows that the page has been last modified on 21 Jul 2017 09:32:54 GMT, the constraint imposed by the conditional GET of request message 2 will not be satisfied.

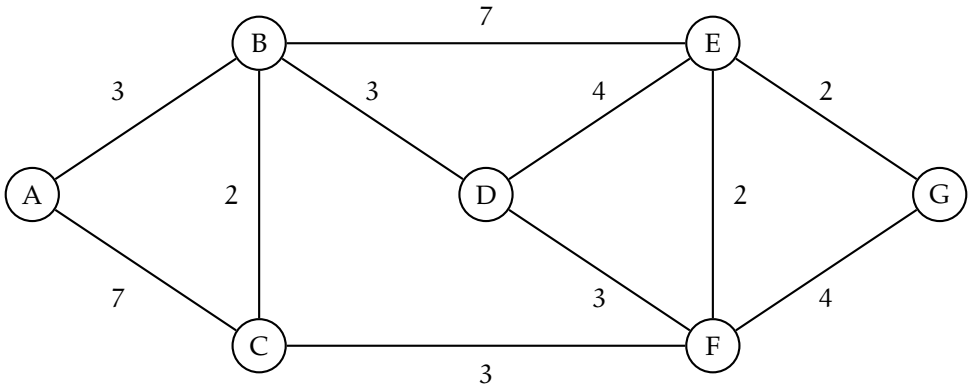
Short Questions, 3.3.3: Eve has successfully compromised the authoritative DNS server of ISP telenor.dk, and poisoned the DNS cache of the server with a malicious record for the personal banking system of mydanishbank.com. Unaware of the cache poisoning, Bob goes to mydanishbank.com on his browser and gets the response from Eve's server. However, Bob knows that mydanishbank.com uses HTTPS in its personal banking page and that his browser should show a lock closed whenever he communicates with the bank through an encrypted channel. Can Eve fool Bob's browser to show a secure connection to Bob with the lock closed? Can Eve decrypt any of the information sent from Bob to her server? Explain why or why not.

Eve cannot fool Bob's browser to show a secure connection to Bob if the lock is closed. Since HTTPS uses SSL/TLS which provides end-point authentication by using digital certificates certified by a CA, the browser will only show the lock closed if the certificate of mydanishbank.com is authentic. Since Eve cannot substitute this certificate, she cannot fool Bob's browser. Note that Eve can use self-signed certificate and if Bob accepts those, then security is compromised (the lock will not closed in that scenario). Since SSL/TLS provides end-point encryption as well by using a session key which is negotiated utilizing the public key of mydanishbank.com, Eve is not privy to the session key and hence cannot cannot decrypt any of the information sent from Bob to her server.

Exam number: _____

3.4 Network Routing (about 18 minutes %)

Consider the network topology outlined in the graph below



Network Routing, 3.4.1: Apply the link state routing algorithm and compute the forwarding tables on nodes A and D. (Note: Remember to show the steps of the algorithm.)

The computation of link state routing algorithm on node A:

Step	N′	D(B),p(B)	D(C),p(C)	D(D),p(D)	D(E),p(E)	D(F),p(F)	D(G),p(G)
0	A	3,A	7,A	∞	∞	∞	∞
1	AB		5,B	6,B	10,B		
2	ABC					8,C	
3	ABCD				10,D/B	8,C	
4	ABCDF				10,D/B/F		12,F
5	ABCDFE						12,F/E
6	ABCDFEG						

In the above computation, if there are multiple paths with same cost they have been shown with / to demarcate them being acceptable values.

The computation of link state algorithm on node D:

Step	N′	D(A),p(A)	D(B),p(B)	D(C),p(C)	D(E),p(E)	D(F),p(F)	D(G),p(G)
0	D	∞	3,D	∞	4,D	3,D	∞
1	DB	6,B		5,B	4,D		
2	DBF			5,B	4,D		7,F
3	DBFE					3,D	6,E
4	DBFEC	6,B					
5	DBFECA						
6	DBFECAG						

In the above computation because of equal cost paths existing the choice of nodes in N′ can be inverted in step 1 and 2 and in steps 5 and 6.

Exam number: _____

Forwarding table on node A follows:

Destination node	Edge
all nodes	(A,B)

Forwarding table on node D follows:

Destination node	Edge
A	(D,B)
B	(D,B)
C	(D,B)
E	(D,E)
F	(D,F)
G	(D,E)

Network Routing, 3.4.2: List the problems that are overcome using hierarchical routing.

Some of the problems that are solved by hierarchical routing are:

1. Scalability issues in message communication overheads with growing network size.

2. Scalability issues in routing algorithm computing overheads with growing network size.

3. Scalability issues in storage and retrieval overheads of routing information with growing network size.

2. Administrative autonomy which allows each autonomous network to operate and administer itself while still being able to inter-operate with other networks.

3.5 Socket Programming (about 15 minutes %)

Socket Programming, 3.5.1: Consider the thread-based concurrent server outlined below that listens on port 5000 and reports back to the client the size (number of bytes) of each received message until the client has closed the connection. The program is **buggy but compiles**. Identify the bugs in the program.

```
#include<unistd.h>
#include<stdlib.h>
#include<sys/socket.h>
#include<arpa/inet.h>
#include<pthread.h>
#include<stdio.h>

void *thread(void *vargp)
{
    int connfd = *((int *)vargp);
    char read_buf[1024], size_buf[sizeof(ssize_t)];
    ssize_t len = read(connfd, read_buf, 1024);
    sprintf(size_buf, "%ld", len);
    write(connfd, size_buf, sizeof(ssize_t));
    return NULL;
}

int main(int argc, char **argv)
{
    int listenfd, connfd;
    socklen_t clientlen = sizeof(struct sockaddr_storage);
    struct sockaddr_storage client_addr;
    struct sockaddr_in server_addr;
    pthread_t tid;

    server_addr.sin_family = AF_INET;
    server_addr.sin_port = htons(5000);
    server_addr.sin_addr.s_addr = htonl(INADDR_ANY);
    listenfd = socket(AF_INET, SOCK_DGRAM, 0);
    if(listenfd != -1) {
        if(bind(listenfd, (struct sockaddr*)&server_addr, sizeof(server_addr)) == 0) {
            listen(listenfd, 10);
            while (1) {
                connfd = accept(listenfd, (struct sockaddr*) &client_addr, &clientlen);
                pthread_create(&tid, NULL, thread, &connfd);
                close(connfd);
            }
        } else {
            close(listenfd);
            exit(EXIT_FAILURE);
        }
    }
    return 0;
}
```

1. Uses SOCK_DGRAM instead of SOCK_STREAM.

2.Does not check the return value of listen.

3. Does not check return value of accept.

4. Does not check return value of pthread_create.

5. Thread resources have not been reaped.

6. Does not deal with the race condition on reading and writing connfd.

7. Closes the connected socket received from accept in the main thread.

8. Does not account for short counts in read and write.

9. Closes socket after receiving one message in the child thread instead of waiting for client to terminate the connection.

10. The size of each individual message read from the connected socket can be a maximum of 1024 bytes (size of read_buf).

Socket Programming, 3.5.2: Compare the pros and cons of designing a concurrent server based on threads and a concurrent event-driven server based on I/O multiplexing.

Thread based servers (TBS) allow for multiple physical control flows by utilizing threads and share the address space between them while event driven servers (EDS) utilize a single logical control flow and multiplex different events in the logical control flow. Their relative comparison follows:

1. Both TBS and EDS make it extremely easy to share data structures. 2. TBS must explicitly handle all concurrency issues that arise in accessing these shared data structures which is not the case in EBS. 3. TBS can take advantage of multiple cores in a machine and leverage fine grained parallelism by utilizing threading construct which is extremely hard if not impossible to do in EBS. 4. Programming EBS is significantly harder than TBS due to the scheduling decisions for handling events. 5. EBS has no overheads from process/thread creation, maintenance and synchronization allowing for higher performance as opposed to TBS. 6. Since EBS has a single flow of control, it is easier to debug using a debugger as opposed to TBS.