State Final Examination (Computer Science Sample Questions)

Spring 2020

1 Automata and grammars (3 points)

1. Formulate Kleene's Theorem. Give definitions of all notions used in the formulation.
2. Prove that for any regular language L, the language

$$L' = \{w \mid \text{every prefix of } w \text{ of odd length belongs to } L\}$$

is also a regular language. (Hint: Show that the complement of the language L' is regular.)

2 Flows in networks (3 points)

1. Define the problem of a maximum flow in a network.
2. Define an augmenting path and use it to characterize a maximum flow in a network.
3. Does the selection of (some) *shortest* augmenting path guarantee that the augmenting path (Ford-Fulkerson) algorithm terminates and runs in polynomial time in all cases?
4. Assume that we are also given maximum capacities for a flow through each vertex in a network. Reduce the problem of finding maximum flow in such a network to the standard problem of maximum flow in a network without constraints for vertices.

3 Databases (3 points)

1. Consider a schedule $T_{12} = X_1(A)$, $W_1(A)$, $U_1(A)$, $X_2(A)$, $R_2(A)$, $U_2(A)$, $S_1(B)$, $R_1(B)$, $U_1(B)$. The notation uses X_i and S_i exclusive and shared lock requests, respectively, of the i-th transaction, U_i denotes unlocking. R_i and W_i correspond to reading from and writing to the specified variable in the i-th transaction. Does the schedule T_{12} satisfy the requirements of the 2-phase (2PL) locking protocol? If not, fix it. If possible, do not change the global order of all the read and write operations included in the schedule.
2. Using the table $Temperature(\underline{Year, Month, Day}, Degrees)$ containing data for the years 2010 to 2019, write a query that enumerates all the months of 2019 that experienced temperatures below zero for at least two days in a row. (Ignore situations where the temperature dropped below zero the very last day of a month and the very first day of the next month.)

4 Object oriented programming and threads (3 points)

1. In the context of the C# program below, explain the difference at the machine code level between the call of the non-virtual method `f` and the call of the virtual method `m`, both inside `Print`. What mechanism is used at runtime to choose whether to execute the code block `B` or `D` during the `x.m` call in `Print`?

```
class X {
    public void f() { A }
    public virtual void m() { B }
}

class Y : X {
```

```
    public new void f() { C }
    public override void m() { D }
}

class Program {
    static void Main() {
        Print(new Y());
    }
    static void Print(X x) {
        x.f();
        x.m();
    }
}
```

(If you prefer the C++ programming language, you can solve this task in an equivalent C++ context.)

2. Write a C++, C# or Java declaration of a function whose task is to sort, in place, an array of objects of some type `T` (the array is one of the function parameters) according to some general criterion – callers of the function must be able to pass an arbitrary sorting criterion as one of its arguments. Also write the declarations of all the necessary types.

   Provide a usage example for the function defined above, assuming we want to sort an array of the `Person` types below. Provide two different usage scenarios: one sorting the array by `Person`'s last name, and one sorting the array by `Person`'s age.

```
class Person {
    public string FirstName;
    public string LastName;
    public int Age;
}
```

3. Write a C++, C# or Java function `MergeSort`, which will take 3 arguments – an array of objects of some type `T`, a description of the sorting criterion (as designed above), and an integer number `N` that is a power of 2. The function should execute parallel sorting of the input array in exactly `N` threads using the *merge sort* algorithm with the given sorting criterion (the final phase of merging the remaining `N` parts of the array may run in less than N threads). You can use a temporary array in your solution. It is not necessary to process partial results or do other sophisticated optimizations, nor to address all possible corner cases of the *merge sort* algorithm – focus on correct evaluation of the sorting criterion and on conceptually correct creation and coordination of multiple threads.

5 Computer architecture (3 points)

Figure 1 shows the structure of an IP packet header. Each row represents one 32-bit word consisting of four bytes. The numbers of the left indicate offset in bytes from the start of the packet. Individual bits are numbered from 0 to 31, with the **most significant bit** (MSb) being **numbered 0** in this case. Multi-byte fields are stored using **big endian** ordering, and the numeric fields contain unsigned integers. The **Header Length** field stores the length of the header in 32-bit words. The **Total Length** field stores the total length of the packet (including the header) in bytes. A packet that is too long can be split into multiple parts—fragments. Fragments are also IP packets (the header is the same), but require interpreting the **Fragment Offset** field, which stores the offset of the data part of the fragment in the data part of the original (non-fragmented) packet. Fragment offset is always aligned to 8 bytes and is stored in the header as a number of 8-byte blocks from the start of the original data part. The **Options** part has a variable length and does not have to be present, because it contains optional fields. The **Padding** is also a variable-length field and pads the header with zeros so that it always ends at 32-bit (4 byte) boundary. The **Data** part contains the packet payload, the format of which depends on the protocol.

Part A

Answer the following questions based on the information about the IP header and its structure:

1. What is the minimal and maximal possible length of a packet header in bytes?
2. What is the minimal and maximal possible length of an entire packet in bytes?
3. What is the maximal possible fragment offset in bytes and which number would represent it in the **Fragment Offset** header field?
4. Can a fragment with the maximal possible value in the **Fragment Offset** field be considered valid? Consider if there could be a valid non-fragmented packet from which such a fragment would be created. Justify your answer.

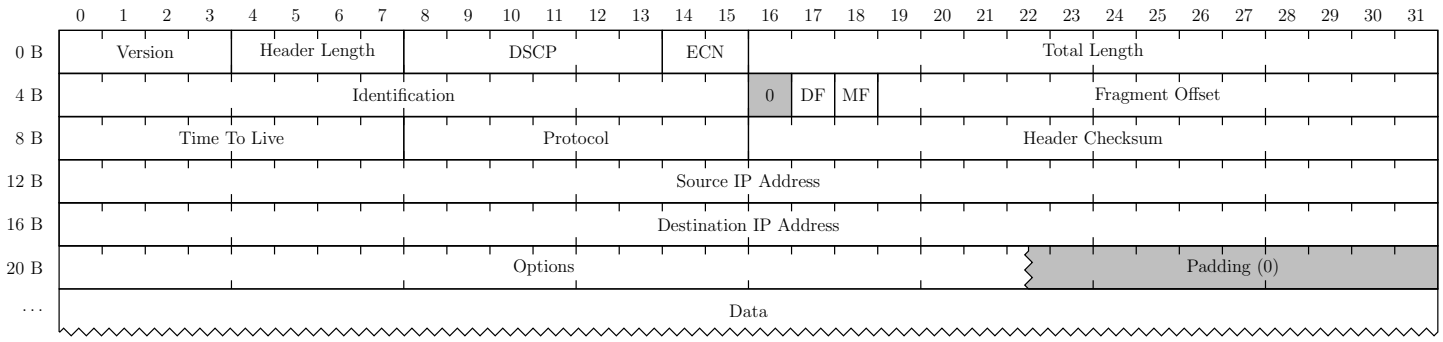

Figure 1: IP header structure

Part B

Figure 2 shows the contents of a memory area containing an IP packet (all values shown are hexadecimal). The IP packet header, the structure of which is shown in Figure 1, starts at address `0xCAFE0010`.

1. At which address starts the data part of the packet and what is the length of the data part in bytes?
2. Does the packet header contain any optional part (**Options**) and padding? If yes, what is the length of the optional part (including the padding) in bytes?

Address	*Memory contents*							
0xCAFE0000	BE	EF	7C	B2	7D	45	FE	9D
0xCAFE0008	DC	02	8E	30	CA	EA	08	00
0xCAFE0010	45	00	00	34	B1	BF	40	00
0xCAFE0018	36	06	06	FA	04	1F	C6	3E
0xCAFE0020	C0	A8	01	05	01	BB	9A	6C
0xCAFE0028	82	6A	E0	E4	5E	31	44	E4
0xCAFE0030	80	10	00	73	43	2B	00	00
0xCAFE0038	01	01	08	0A	F4	DF	51	5B
0xCAFE0040	1C	30	A2	1C	FE	31	1F	C0

Figure 2: Contents of memory holding an IP packet

Part C

Assume that a router uses a simple 32-bit RISC processor with load/store architecture. The processor has 16 general-purpose 32-bit registers named `$r0`–`$r15`, with register `$r0` always containing the value of `0`. The table in Figure 3 provides a brief overview of basic instructions of the processor. In the description, we use *imm32* for 32-bit constants and *label* for a symbolic replacement of an address in the code. Memory addressing is byte-oriented, but the processor can only access **32-bit words** at aligned addresses, i.e., addresses **divisible by 4**. When accessing the memory, the processor uses the **little-endian** byte ordering.

Memory access		
lw $rd, *imm32* ($rs)	Load 32-bit word to register	R[rd] = Memory[R[rs] + *imm32*]
sw $rs, *imm32* ($rd)	Store 32-bit word to memory	Memory[R[rd] + *imm32*] = R[rs]
Arithmetic and logic operations		
add/sub $rd, $rs, $rt	Add/subtract registers	R[rd] = R[rs] ± R[rt]
addi/subi $rd, $rs, *imm32*	Add/subtract register and immediate	R[rd] = R[rs] ± *imm32*
and/or/xor $rd, $rs, $rt	Bitwise and/or/xor with register	R[rd] = R[rs] *op* R[rt]
andi/ori/xori $rd, $rs, *imm32*	Bitwise and/or/xor with immediate	R[rd] = R[rs] *op* *imm32*
sll/slr/sra $rd, $rs, $rt	Shift logical left/right, right arithmetic	R[rd] = R[rs] *op* (R[rt] **mod** 32)
slli/slri/srai $rd, $rs, *imm32*	Shift logical left/right, right arithmetic	R[rd] = R[rs] *op* (*imm32* **mod** 32)
li $rd, *imm32*	Load immediate to register	R[rd] = *imm32*
move $rd, $rs	Move (copy) value between registers	R[rd] = R[rs]
Comparison and conditional branches		
slt $rd, $rs, $rt	Signed *set on less than* betwen registers	**if** (R[rs] $<_{signed}$ R[rt]) **then** R[rd] = 1 **else** R[rd] = 0
sltu $rd, $rs, $rt	Unsigned *set on less than* between registers	**if** (R[rs] $<_{unsigned}$ R[rt]) **then** R[rd] = 1 **else** R[rd] = 0
slti $rd, $rs, *imm32*	Signed *set on less than* with immediate	**if** (R[rs] $<_{signed}$ *imm32*) **then** R[rd] = 1 **else** R[rd] = 0
sltiu $rd, $rs, *imm32*	Unsigned *set on less than* with immediate	**if** (R[rs] $<_{unsigned}$ *imm32*) **then** R[rd] = 1 **else** R[rd] = 0
beq $rs, $rt, *label*	Branch on registers equal	**if** (R[rs] == R[rt]) **then** PC = *label* (address)
bne $rs, $rt, *label*	Branch on registers not equal	**if** (R[rs] != R[rt]) **then** PC = *label* (address)
Unconditional jumps		
j *label*	Jump to address (*label*)	PC = *label* (address)
jr $rs	Jump to address in register	PC = R[rs]

Figure 3: Basic instructions of the processor

Given the information about the IP header and its structure (Figure 1), the information about the processor architecture limitations, and the provided instruction set (Figure 3), write a fragment of code that decrements the value of the **Time To Live** (TTL) field by `1`, but **only if it is non-zero** (to prevent underflow). Assume that register `$r1` contains the address of the IP packet, and that the address is aligned (to a 4-byte boundary).

6 TCP/IP architecture (3 points)

Your network traffic monitoring tool captured the following packet:

```
0000   00 1f 1f c0 ca 33 54 e1 ad 15 39 92 08 00 45 00   .....3T...9...E.
0010   00 5c 21 f2 40 00 40 06 df f9 c0 a8 00 10 5f d8   .\!.@.@........_.
0020   18 20 8f e6 00 50 c7 f5 f6 7d f8 77 22 8c 80 18   . ...P...}.w"...
0030   01 f6 34 3d 00 00 01 01 08 0a 30 c9 1c 8b 9b 1a   ..4=......0.....
0040   19 29 47 45 54 20 2f 20 48 54 54 50 2f 31 2e 30   .)GET / HTTP/1.0
0050   0d 0a 48 6f 73 74 3a 20 77 77 77 2e 61 70 61 63   ..Host: www.apac
0060   68 65 2e 6f 72 67 0d 0a 0d 0a                     he.org....
```

1. The captured traffic uses the IP protocol with source address 192.168.0.16 (the IP packet header structure was given in the previous question). In the dump, delineate the link layer data and the internet layer data. What information is probably contained in the link layer data ?

2. Based on the packet content, also estimate the location of the transport layer data and the application layer data, and explain your estimate (an exact answer would require knowing other standard protocol structures, however, an informed guess can work quite well in this case). What is the size of the data at the individual layers (all four) in bytes ?

3. Also estimate what applications are communicating with each other using this packet, and what other protocols in addition to the IP protocol are used. Explain your answer.

4. Assume the communicating applications need to transfer a data block larger than what the network equipment on the path between the sender and the recipient can transport in one piece (for example the sender would request sending a 1 MB data block using the `write` syscall on the corresponding network connection). In this situation, where would the block be split and recombined ?

7 Set theory (specialization question – 3 points)

1. Give the definition of strict well-ordering. It is not necessary to define the notion of ordering itself.
2. Is the set $\mathbb{Q} \cap [0, \infty)$ of nonnegative rational numbers strictly well-ordered by the standard relation "$<$" ? Explain.
3. Give the definition of an ordinal number.

8 Ramsey theory (specialization question – 3 points)

1. State the Ramsey theorem (for finite graphs, with k colors).
2. Prove that there exists an integer n such that the following claim holds. Suppose S is a set of n axis–aligned rectangles (not necessarily of the same size) in the plane. If every point of the plane is contained in at most two of the rectangles, then S contains a subset of 100 pairwise disjoint rectangles.
3. Prove that every triangle–free graph with n vertices contains an independent set of size at least $\sqrt{n} - 1$.

9 Perfect codes (specialization question – 3 points)

1. What does it mean that a binary error-correcting code is perfect?
2. Determine the minimum distance of the linear code over $\mathbb{Z}_2$ generated by vectors $(0, 0, 1, 0, 1, 1), (1, 0, 0, 1, 0, 1)$, and $(1, 1, 0, 0, 1, 1)$. Is this code perfect?