Qual Exam Note

Sangwoo Lee

June, 2023

1 Introduction

I started my preparation for the qualifying exam with absolutely no background knowledge in the relevant subjects. I had not taken any of the fundamental courses that are usually needed for this exam, such as operating systems or computer architecture. In fact, the only related class I had taken before the exam was an algorithms class. Despite this, I was able to pass the exam because I focused on understanding the practice questions and learning the techniques to solve them. Based on my experience, I would say that for someone preparing for the qualifying exam, it is crucial to work on the practice questions and understand the solutions.

You can find a set of practice questions on the department's webpage. I would recommend working on the first three sets of practice questions, as each of these sets contains 70 questions. The fourth set has significantly fewer, with only 13 questions. I was able to find solutions for the second and third sets of practice questions, but not for the first set. Because of this, I went ahead and created solutions for the first set myself. I have included these, along with the solutions for the second and third sets, in this document. I really hope that this material will be of assistance to you all!

2 CS GRE Practice 1

2.1 Question 1

$\frac{10000}{10000+20} = 99.8888....$ Please do not calculate exact number. Instead, you can guess that $\frac{10000}{10000+20}$ is is higher than 98% ($\frac{9800}{10000}$).

2.2 Question 2

Recursion existed before the object-oriented programming language. For example, C (not C++) has a recursion feature. Thus, the answer is (c).

2.3 Question 3

Quicksort has an average running time of $O(n \log n)$. However, it takes $O(n^2)$ running time if the elements are already sorted. Thus, the answer is (d).

2.4 Question 4

1. **Symbol table**: It keeps track of semantic information about identifiers (e.g., variables and function names). Specifically, it stores the type of symbols (integer, float, function) and their scope (local or global).

2. **Abstract syntax tree**: This is a tree representation of the abstract syntactic structure of the source code. It is built by a parser during the syntax analysis.

3. **Parser table**: This is a two-dimensional array used by the parser in the syntax analysis phase. The rows represent states the parser can be in, and the columns represent input symbols. The type of parser determines the construction of this table.

4. **Intermediate code**: The compiler often generates an intermediate code representation of the source code. This is used as a stepping stone to generate the final machine code. It can be represented in various forms, such as three-address code, syntax trees, or quadruples.
5. **Control flow graph (CFG)**: It is used in the optimization phase. Each node represents a basic block, a straight-line piece of code without any branches. Edges represent control flow (jumps, branches) between these blocks. This structure helps in various optimizations like dead code elimination, loop optimization, etc.
6. **Call Graph**: It is a directed graph that represents calling relationships between functions. Each node is a function. Each edge (f, g) indicates that function f calls function g.
7. **Actication record or Stack frame**: It contains information about the runtime state of the procedure or subroutine. Specifically, it includes information like return address, passed parameters, local variables, and temporary variables. The compiler uses this information to manage function calls and returns.

Thus, the answer is (c).

2.5 Question 5

The loop ends if x is larger than 1000.

1. 1st iteration: It starts with $i = 1$. Then, x becomes $2^1 = 2$, and i becomes 2.
2. 2nd iteration: It starts with $i = 2$. Then, x becomes $2^2 = 4$, and i becomes 3.
3. 3rd iteration: It starts with $i = 3$. Then, x becomes $2^4 = 16$, and i becomes 4.
4. 4th iteration: It starts with $i = 4$. Then, x becomes 2^{16}, and i becomes 5.

After the 4th iteration, the loop ends because 2^{16} is larger than 1000 ($2^{10} = 1024$). Thus, the answer is (b).

2.6 Question 6

- x is a parent of y and v. And, they are different people, which means y and v are siblings.
- y is a parent of w.
- v is male.

Thus, v is an uncle of w.

2.7 Question 7 and 8

You can find an answer if you know the below.

- **Preorder**: Parent → Left Child → Right Child
- **Postorder**: Left Child → Right Child → Parent
- **Inorder**: Left Child → Parent → Right Child

2.8 Question 9

The answer is (a).

- (A): Circuit switching was mainly used for wired phone call services.
- (B): Depending on topology and how the vendors constructed the infrastructure. It can be true.
- (C): CSMA/CD is a de facto MAC protocol for wired networks.
- (D): I do not think this is true nowadays. But, I think this was true like 20 years ago.
- (E): Every network protocol has a maximum packet size.

2.9 Question 10

I think the easiest way to solve this problem is by drawing a simple example. I drew 3-ary tree with a height of 2 and figured out that the maximum number of leaves are 9 which is 3^2. Thus, the answer is (D).

2.10 Question 11

- (I): This is true. Virtual memory allows a computer to execute programs and manipulate data sets that are larger than its physical RAM.
- (II): This is not true. It is possible for parts of programs to be moved in and out of the main memory (RAM) as needed during execution. This technique is known as swapping or paging.
- (III): This is not true. The division into pages is done for the sake of memory management efficiency and does not take into account the logical structure or semantic characteristics of the program.

Thus, the answer is (a).

2.11 Question 12

In some questions, you may need to figure out the order of growth, given the execution time $T(n)$ where n is the number of inputs. Basically, you need to unravel the recurrence relation by organizing the expression for $T(n)$. However, that is time-consuming in general. Thus, I prefer to use some tricks for quick solving.

- $\Theta(n)$ shows the pattern of $1 + 2 + 4 + 8 + 16 + \cdots + n$. It increases power of 2.
- $\Theta(n^2)$ shows the pattern of $1 + 2 + 3 + \cdots + n$. It increases by one.
- $\Theta(k^n)$ shows the pattern of $1 + k + k^2 + k^3 \cdots + k^n$. It increases power of k.
- $\Theta(log_2 n)$ rarely appears in the exam. In general, if while loop shows the following pattern, then the order of growth is $\Theta(log_2 n)$: **While**$(i < N)$ $\{i = i*2\}$; Please note that the loop is executed once more if N is doubled.

2.12 Address resolution protocol

ARP protocol is used by IPv4 to map IP address to the hardware address in the link layer. In other words, ARP translates 32-bit IP addresses to 48-bit Ethernet MAC addresses. These two sentences are enough for the exam.

2.13 Operations with eight registers (Q.17)

Question 17 made me confused. I had to consider the case of calculating $AB + BC$ and then $(AB + BC) + ABC$ even though the given operations are $AB + ABC + BC$. Figure 2 shows how I found the correct answer. Please note that you should take care of the order of operations and the space of the registers.

12. Let $T(n)$ be defined by $T(1) = 7$ and $T(n+1) = 3n + T(n)$ for all integers $n \geq 1$. Which of the following represents the order of growth of $T(n)$ as a function of n ?

(A) $\Theta(n)$ (B) $\Theta(n \log n)$ (C) $\Theta(n^2)$ (D) $\Theta(n^2 \log n)$ (E) $\Theta(2^n)$

$T(n+1) = 3n + T(n)$

$= 3n + 3(n-1) + 3(n-2) + T(n-2)$

$= 3\{n + (n-1) + (n-2) + \cdots 1\} + T(1)$

↳ This pattern is "$\Theta(n^2)$". So, we can know the answer of question at this point.

$= 3\{\frac{n}{2} \cdot (n+1)\} + T(1)$

↳ sum of an arithmatic series.

Sum = $\frac{n}{2}$ (first term + last term)

$= 3\{\frac{1}{2}n^2 + \frac{1}{2}\} + 7$

↳ $\Theta(n^2)$

Figure 1: Question 12

17. A certain pipelined RISC machine has 8 general-purpose registers `R0`, `R1`, . . . , `R7` and supports the following operations.

`ADD Rs1, Rs2, Rd`	Add `Rs1` to `Rs2` and put the sum in `Rd`
`MUL Rs1, Rs2, Rd`	Multiply `Rs1` by `Rs2` and put the product in `Rd`

An operation normally takes one cycle; however, an operation takes two cycles if it produces a result required by the immediately following operation in an operation sequence. Consider the expression $AB + ABC + BC$, where variables A, B, C are located in registers `R0`, `R1`, `R2`. If the contents of these three registers must not be modified, what is the minimum number of clock cycles required for an operation sequence that computes the value of $AB + ABC + BC$?

(A) 5 (B) 6 (C) 7 (D) 8 (E) 9

R_0	R_1	R_2	R_3	R_4	R_5	R_6	R_7
A	B	C	AB	BC	ABC	AB+BC	AB+ABC+BC

① $R_0 * R_1 \Rightarrow R_3 \Rightarrow 1$

② $R_1 * R_2 \Rightarrow R_4 \Rightarrow 1$

③ $R_3 * R_2 \Rightarrow R_5 \Rightarrow 1$

④ $R_3 + R_4 \Rightarrow R_6 \Rightarrow 2$

⑤ $R_6 + R_5 \Rightarrow R_7 \Rightarrow 1$

Total = 6

Figure 2: Question 17

2.14 Cache Update

1. **Write-through**: Every write operation to the cache is also written to the main memory (RAM). The data in the main memory is always in sync with the data in the cache.

2. **Write-back**: Write operations are only carried out on the cache. Then, the main memory is updated after the termination of the program or cache abort.

2.15 Predicate calculus formulas

Figure 3 shows Question 23. I recommend that you make a sentence on each calculus formula and then think about it intuitively. We do not have enough time during the exam, and there are no complicated calculus formulas in this exam.

23. Which of the following predicate calculus formulas must be true under all interpretations?

I. $\Big(\forall x P(x) \vee \forall x Q(x)\Big) \rightarrow \forall x\Big(P(x) \vee Q(x)\Big)$

II. $\forall x\Big(P(x) \vee Q(x)\Big) \rightarrow \Big(\forall x P(x) \vee \forall x Q(x)\Big)$

III. $\Big(\exists x P(x) \vee \exists x Q(x)\Big) \rightarrow \exists x\Big(P(x) \vee Q(x)\Big)$

(A) I only (B) III only (C) I and II (D) I and III (E) II and III

Figure 3: Question 23

1. $\forall x P(x) \vee \forall x Q(x)$, this statement says $P(x)$ is true for all x or $Q(x)$ is true for all x.

   $\forall x(P(x) \vee Q(x))$, this statement says, for all x, $P(x)$ is true or $Q(x)$ is true.

   For your understanding, consider a situation where $P(x)$ is true for all x, but $Q(x)$ is not true for all x. The first statement $\forall x P(x) \vee \forall x Q(x)$ would be true because $\forall x P(x)$ is true. The second statement $\forall x(P(x) \vee Q(x))$ would be also true because $P(x)$ is always true even if $Q(x)$ is not true. Thus, this calculus formula is always true.

2. This is the opposite case of the first one.

   Again, $\forall x(P(x) \vee Q(x))$ says, for all x, $P(x)$ is true or $Q(x)$ is true. However, it does not imply the right statement $\forall x P(x) \vee \forall x Q(x)$ is always true. Thus, it is not always true.

   For your understanding, consider a situation where some x satisfy $P(x)$ and some other x satisfy $Q(x)$, but it's not the case that all x satisfy $P(x)$ or that all x satisfy $Q(x)$.

3. $\exists x P(x) \vee \exists x Q(x)$ says $P(x)$ is true for some x or $Q(x)$ is true for some x.

   $\exists x(P(x) \vee Q(x))$ says, for some x, $P(x)$ is true or $Q(x)$ is true.

   If $\exists x P(x) \vee \exists x Q(x)$ is true, there must be x that makes either $P(x)$ true or $Q(x)$ is true. Thus, it makes always $\exists x(P(x) \vee Q(x))$ true.

2.16 Formal Languages Basic

Figure 4 shows Question 26 which asks you for the basic knowledge of formal languages.

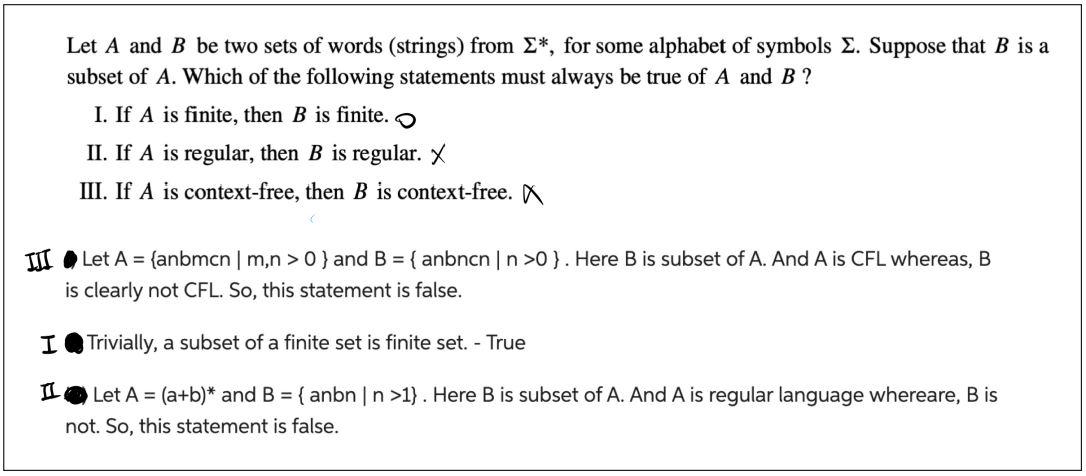


Figure 4: Question 26

I have not taken automata or compiler classes. So, I had to study formal language and automata myself. I would not write down what I studied here. Instead, I would like to give you some keywords that you may want to study. There are a bunch of good blogs or youtube videos for learning (I mainly used Korean blogs).

- Concept of formal language (symbol, alphabet, Kleene Star, and so on).
- Deterministic Finite Automata (DFA), Non-deterministic Finite Automata (NFA), regular language, pumping lemma.
- Context Free Language (CFL), Context Free Grammar (CFG)

2.17 Carry-out and overflow

- **Carry-out**: A carry-out occurs when the result of an addition operation on the most significant bits of the numbers exceeds the capacity of that bit.
- **Overflow**: Overflow is a condition that happens when the result of an operation is larger than the maximum value that can be stored.

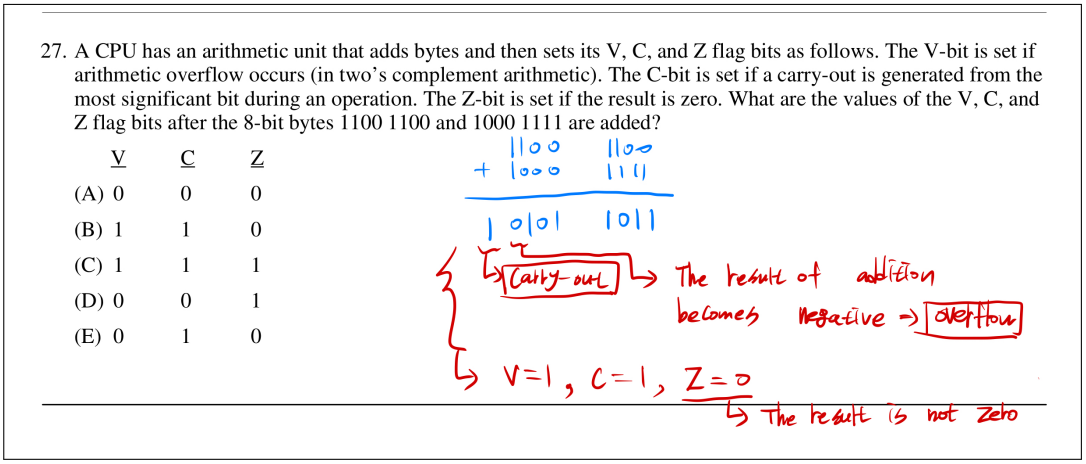


Figure 5: Question 27

2.18 Merge Sort

The complexity of merge sort is not affected by data type as all n components should be compared on every merging step.

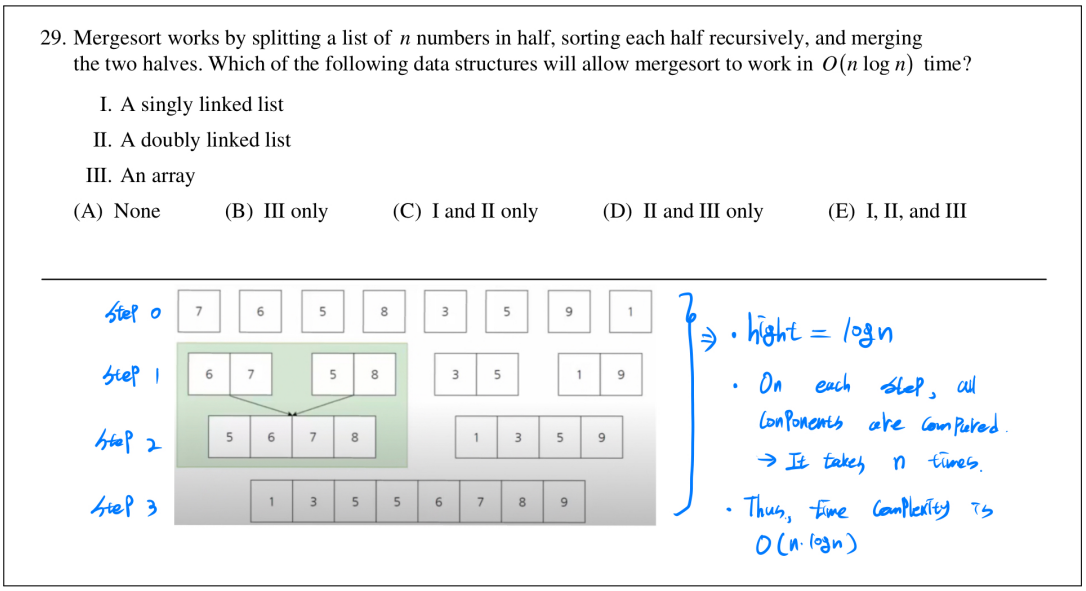


Figure 6: Question 29

2.19 NP-hard Problems (Q. 32)

You need to remember the well-known NP-hard problems.

- **Travelling Salesman Problem (TSP)**: The problem is to find the shortest possible route through a set of cities, visiting each city exactly once, and returning to the start city.
- **Knapsack Problem**: Given a set of items, each with a weight and a value, determine the items to include in a collection so that the total weight is less than or equal to a given limit and the total value is as large as possible.
- **Integer Programming**: This is a mathematical optimization or feasibility problem in which some or all of the variables are restricted to be integers.
- **Graph Coloring Problem**: Assign colors to vertices in a graph such that no two adjacent vertices share the same color. The goal is to minimize the number of colors while satisfying this condition.
- **Bin Packing Problem**: Given objects of different volumes, find the smallest number of bins that can contain them all, given a bin volume limit.
- **Vehicle Routing Problem**: Like the traveling salesman problem, but with multiple vehicles each starting from a central depot.
- **Clique Problem**: A clique in a graph is a subset of vertices such that every two vertices in the subset are connected by an edge. The Clique Problem is to determine whether a graph contains a clique of a given size.
- **Hamiltonian Cycle Problem**: Is there a cycle in an undirected graph that visits each vertex exactly once? A cycle that meets these criteria is called a Hamiltonian cycle.
- **Vertex Cover Problem**: Given an undirected graph, the task is to find a minimum size subset of vertices such that each edge of the graph is incident to at least one vertex of the subset.

- **Steiner Tree Problem**: Given a graph and a subset of vertices in the graph, the problem is to find the shortest possible tree or trees that cover all the vertices in the subset.
- **Longest Cycle Problem (LCP)**: Given a graph, find the longest simple cycle (i.e., a cycle that does not repeat vertices) in the graph.
- **Finding ALL spanning trees**: The time complexity of finding all spanning trees is dependent on the number of spanning trees, which can be very large. Thus, the task of finding all spanning trees cannot be accomplished in polynomial time.

2.20 Performance of Processors

Please remember that only a clock cycle affects the maximum throughput of processors.

33. Two processors, M-5 and M-7, implement the same instruction set. Processor M-5 uses a 5-stage pipeline and a clock cycle of 10 nanoseconds. Processor M-7 uses a 7-stage pipeline and a clock cycle of 7.5 nanoseconds. Which of the following is (are) true?

    I. M-7's pipeline has better maximum throughput than M-5's pipeline.

    II. The latency of a single instruction is shorter on M-7's pipeline than on M-5's pipeline.

    III. Programs executing on M-7 will always run faster than programs executing on M-5.

    (A) I only (B) II only (C) I and III only (D) II and III only (E) I, II, and III

Figure 7: Question 33

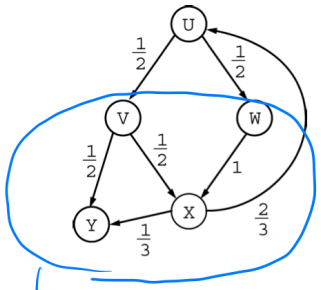
1. M-7 has a shorter clock cycle than that of M-5. Thus, M-7 has better maximum throughput.
2. **M-7**: $7 * 7.5ns = 52.5ns$ / **M-5**: $10 * 5ns = 5ns$. Thus, M-5 has a shorter latency of a single instruction.
3. Execution time depends on programs. Even though M-7 shows a shorter maximum throughput, M-5 can execute some programs faster than M-7. Literally, it depends on the programs.

2.21 Probability of execution

I was confused when I solve Question 34. At first, U is executed for sure. Then, the probability that U is executed once more is $(\frac{1}{2} * \frac{1}{2} * \frac{2}{3}) + (\frac{1}{2} * \frac{2}{3}) = \frac{1}{2}$. Thus, the expected number of times that U executes is $(1 + \frac{1}{2} + \frac{1}{4} + \frac{1}{8} + \dots) \simeq 2$.

34. For the following code, the bias of each conditional branch in the code is labeled on the control flow graph to the right. For example, the Boolean expression `if_condition` evaluates to `true` on one-half of the executions of that expression.

```
do
{
  U;
  if (if_condition)
  {
    V;
    if (break_condition)
      break;
  }
  else
    W;
  X;
} while (loop_condition);
Y;
```



What is the expected number of times that `U` executes?

(A) 0.5

(B) 1

(C) 1.5

(D) 2

(E) More than 10

Figure 8: Question 34

2.22 Karnaugh map

You should study Karnaugh map to solve Question 36.

36. A logic circuit has three input bits: x_0, x_1, and x_2, where x_0 is the least significant bit and x_2 is the most significant bit. The output from the circuit is 1 when its input is any of the 3-bit numbers 1, 4, 5, or 6; otherwise, the output is 0. Which of the following expressions represents the output from this circuit?

(A) $\overline{x_2} + \overline{x_1} + \overline{x_0}$

(B) $\overline{x_2}x_0 + x_2\overline{x_1}$

(C) $\overline{x_1}x_0 + x_2\overline{x_0}$

(D) $\overline{x_2}x_1x_0 + x_2\overline{x_1}$

(E) $x_2 + \overline{x_1}x_0$

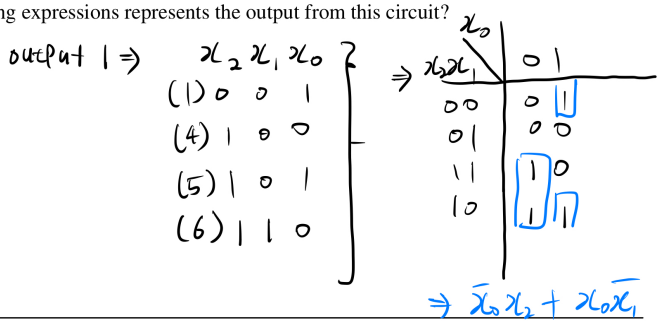


Figure 9: Question 36

2.23 Time Complexities of Well-known Algorithms

- Kruskal: $O(ElogE)$ → max/min spanning tree
- Dijkstra: $O((V+E)logV)$ or $O(VlogV+E)$ with heap → Shortest Path Problem (SPP) in an undirected graph.
- Bellman-Ford: $O(VE)$ → SPP (Negative edge O / cycle with a negative edge X).
- Floyd-warshall: $O(V^3)$ → Shortest paths for all pairs (Negative edge O / cycle with a negative edge X).

Plus, it is good to remember Kruskal, Dijkstra, and Prims are greedy algorithms.

2.24 Root Set of Garbage Collector (Q. 40)

Typically, the root set of garbage collector includes:

- Local variables and input parameters of the currently executing methods
- Data on registers.
- Global and Static variables.
- Call stack.
- References used in a program.

Plus, dynamically allocated objects on the heap are not considered part of the root set because they're not the starting points for the Garbage Collector (GC)'s reachability analysis. Instead, they're the objects that the GC is trying to evaluate for reachability based on the root set.

2.25 Q.42

Please remember that we cannot count the exact number of true variables with connectives.

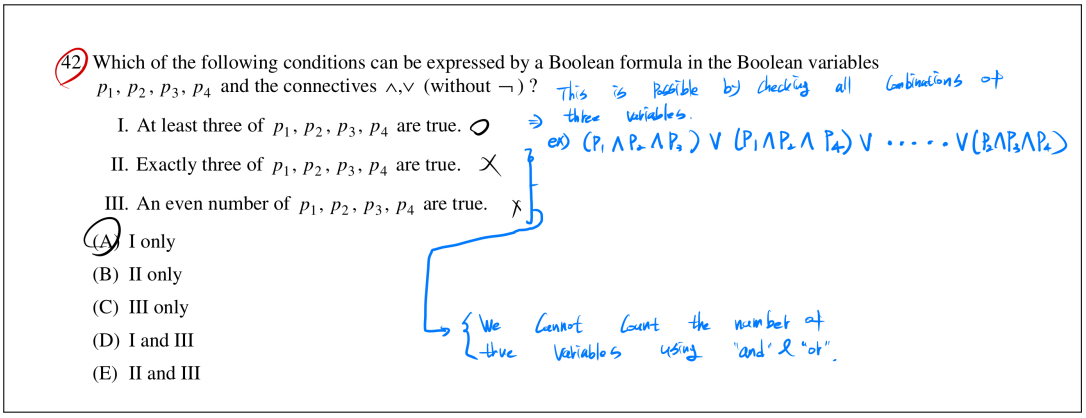


Figure 10: Question 42

2.26 Q.43

Considering that the edges are only 6, we can draw each case.

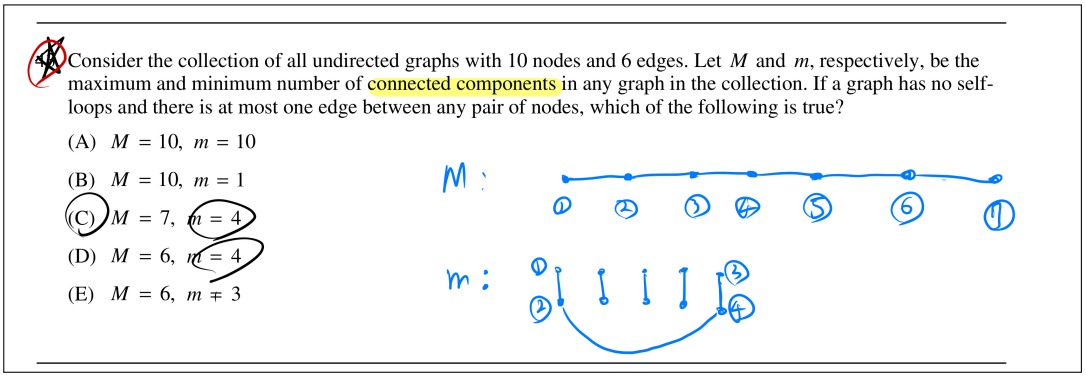


Figure 11: Question 43

2.27 Newton's method (Q.45)

Newton's method is an algorithm for finding successively better approximations to the roots (or zeroes) of a real-valued function. It solves equations of the form $f(x) = 0$, where f is a differentiable function. The high-level description of Newton's method is as follows:

1. Start with an initial guess x_0.
2. If $f(x_0) = 0$, then $x0$ is a root of the function, so return x_0. Otherwise, calculate the next guess $x_1 = x_0 - \frac{f(x_0)}{f'(x_0)}$, where $f'(x_0)$ is the derivative of f at x_0.
3. Repeat the process with x_1 instead of x_0, and so on, until you get a value of x for which $f(x)$ is close enough to 0.

It may be helpful to understand if you see the figure below.

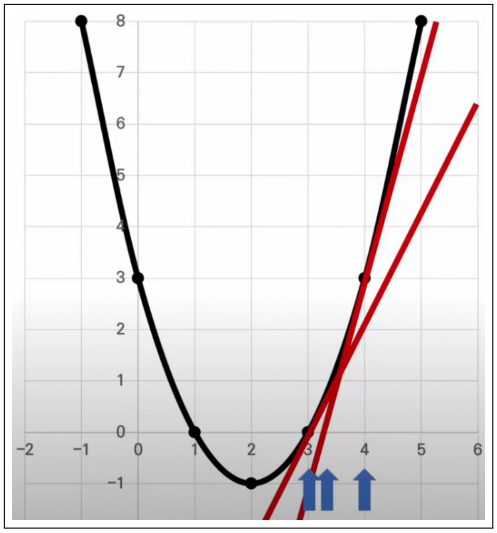


Figure 12: Newton's method example

Most importantly, you need to repeat $\log_2 b$ times to get the approximation that is as same as b digits with the real root. Thus, the order of growth mentioned in Q.45 is $O(\log b)$

2.28 Halting Problem Q.47

Q.47 asks about Deterministic Turing Machine (DTM). If you studied what is "halting problem", it is easy to solve. Simply, the halting problem means that it is not possible to determine whether the program finishes running or will run forever, given that input.

47. Let M be a single-tape, deterministic Turing machine with tape alphabet {*blank*,0,1}, and let C denote the (possibly infinite) computation of M starting with a blank tape. The input to each problem below is M, together with a positive integer n. Which of the following problems is (are) decidable?

    I. The computation C lasts for at least n steps.

    II. The computation C lasts for at least n steps, and M prints a 1 at some point after the nth step.

    III. M scans at least n distinct tape squares during the computation C.

    (A) None

    (B) III only

    (C) I and II only

    (D) I and III only

    (E) I, II, and III

Figure 13: Question 47

1. You can construct a Turing Machine that simulates M and counts the steps. If it reaches n steps, it halts and accepts. Thus, it is decidable.

2. *M prints a 1 at some point after the nth step*: This sentence is equivalent to asking whether M halts after n steps and eventually prints a 1. This falls into the halting problem. Thus, it is not decidable.

3. This problem is decidable. A Turing Machine can simulate M and keep track of the number of distinct tape squares scanned. If M scans n distinct tape squares, the simulator halts and accepts. If M halts before scanning n distinct squares, the simulator halts and rejects.

3 CS GRE Practice 2

The solution file already exists. I attached it from the next page. **I think this set is the oldest practice set. So, I recommend you solve this set after you solve the first and third sets.**

4 CS GRE Practice 3

There are solutions on Youtube. Click here to visit the playlist

Questions and Answers
in
Computer Science

By
Yonas Tesfazghi Weldeselassie
http://www.ictp.trieste.it/~weldesel

The Questions in this material are collected from Computer Science Subject
Graduate Record Examination Preparation Materials

The solutions are explicitly personal and are primarily intended for discussion
purposes with people who want to share ideas

Neither correctness nor completeness is assumed or implied
Do not rely on it

Your feedbacks are most welcome!

Question 1.

Any set of Boolean operators that is sufficient to represent all Boolean expressions is said to be complete. Which of the following is NOT complete?

A. {AND, NOT}

B. {NOT, OR}

C. {AND, OR}

D. {NAND}

E. {NOR}

Solution 1.

Recall that the basics of Boolean algebra operators are AND, OR and NOT. These three operators are needed to express any Boolean algebra. So we need to check which of the above alternative sets fails to express these operators.

To that end, recall that using AND and NOT, we can express OR, using De'Moivre Theorem, as A OR B = $\overline{\overline{A}\ AND\ \overline{B}}$. Similarly, using OR and NOT, we can express AND as A AND B = $\overline{\overline{A}\ OR\ \overline{B}}$. Moreover, NAND can express NOT as the NAND of a boolean variable with itself. Since NAND is negation of AND, it follows that NAND can express AND by simply inverting the result of NAND (i.e. given A and B, A AND B = Negation(A NAND B) = (A NAND B) NAND (A NAND B)). Since it can express AND and NOT, it follows that it can express OR. A similar argument can be applied to NOR. NOR expresses NOT as the NOR of a variable with itself. It follows that it can express OR as the NOR of a result of NOR with itself. Finally De'Moivre's Theorem yields the expression to express AND using NOR (i.e. NOT and OR).

With AND and OR, we can't express NOT in anyway. Thus the answer is **C**.

Question 2.

Which of the following decimal numbers has an exact representation in binary notation?

A. 0.1

B. 0.2

C. 0.3

D. 0.4

E. 0.5

Solution 2.

The only thing we need to remember here is on how to convert decimal numbers to binary.
$\rightsquigarrow$In order to convert integer decimal numbers to binary, divide the number by 2, note the remainder and continue the same operation with the quotient until a quotient of 0 is obtained at which step we need to stop. All the remainders are then written starting from the last to the first resulting the required binary expression.
$\rightsquigarrow$In order to convert a fraction part, as is the case in this question, multiply the fraction by 2, note the integer part of the product, and continue the same operation with the fraction part of the product until a product of 1.0 is obtained. The required binary fraction is then $0.i_1i_2i_3...$ where the i's are the integer parts of the products starting from the first to the last (if the operation stops at some stage).

Apply this rule to 0.1 to get the following products 0.2, 0.4, 0.8, 1.6, (take away 1 and continue with 0.6), 1.2, 0.2, ... Stop! We had already encountered 0.2 which means we will encounter it again and again if we continue which means this operation will go indefinitely. Thus all the choices A, B, C and D do not have finite (exact) binary notation because all the decimal numbers 0.1, 0.2, 0.3 (which will be 0.6 on the second stage) and 0.4 are encountered in the conversion procedure for 0.1 which did not terminate. The only choice left is E; indeed 0.5 will give 1.0 in the second stage which is the terminating stage resulting the binary 0.1 as its binary representation. Hence the answer is choice E.

Question 3.
Bob writes down a number between 1 and 1,000. Mary must identify that number by asking "yes/no" questions to Bob. Mary knows that Bob always tells the truth. If Mary uses an optimal strategy, then she will determine the answer at the end of exactly how many questions in the worst case?

A. 1,000
B. 999
C. 500
D. 32
E. 10

Solution 3.
The difficult part on this question is to understand what sort of questions Mary should ask Bob. If she asks questions like *'Is it the number 764?', 'Is it the number 342?', etc*, then surely Mary will need to ask 999 questions in the worst case in order to get the answer.

The optimal strategy is to ask questions like *'Is it GREATER THAN 500?', 'Is it GREATER THAN 250?', etc*; each time throwing away half of the interval and working with the remaining half. This is simply binary search method and Mary will need to ask a maximum of $log_2(1000)$ questions. Hence the answer is choice E.

Question 4.

If x is a string then x^R denotes the reversal of x. If x and y are strings, then $(xy)^R =$

A. xy^R

B. yx^R

C. $y^R x$

D. $x^R y^R$

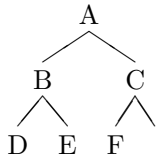
E. $y^R x^R$

Solution 4.

The answer can easily be verified to be choice E. [It is similar to transposing a product two matrices which is the product of the transposes in reverse order.]

Question 5.

A procedure that printed the binary tree



in postorder would produce as output[1]

A. ABCDEF

B. ABDECF

C. DBEAFC

D. DEBFCA

E. DEFBCA

[1]Please note that in this tree diagram and the subsequent tree diagrams, some EDGES are drawn with no actual nodes. This is due to the fact that, according to the package I am using to draw trees (parsetree.sty), if one wants to have only one child node, then the node is drawn as middle child (neither right child not left one). So in order to clarify that indeed the child node is either left or right, I am adding additional empty redundant edge. Such edges should be treated as NON-EXISTENT.

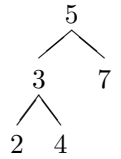
Solution 5.
Just remember that

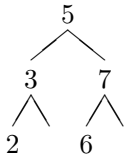
- Inorder traverses a tree from Left to Root and then to Right.
- Preorder traverses a tree from Root to Left and then to Right.
- Postorder traverses a tree from Left to Right and then to Root.

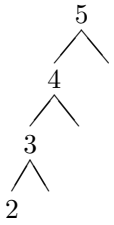
All these traversals start at the root node of the tree and perform the operation on each node of the tree recursively.

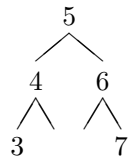
Thus choices A and E refer to nothing, choice B is a result of preorder traversal, choice C is a result of inorder traversal, and choice D is a result of postorder traversal. Hence the answer is choice D.

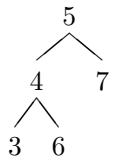
Question 6.
Which of the following is not a binary search tree?

A.


B.


C.


D.


E.


Solution 6.
A binary search tree is a binary tree such that for all the nodes in the tree, we have all left children are less than the root which is less than or equal to all the right children. Such trees are used for searching in logarithmic scale time. The answer is choice E.

Question 7.
Consider the following Pascal-like program fragment.

```
var i, j : integer;
procedure P(k, m : integer);
begin
    k := k - m;
    m := k + m;
    k := m - k;
end
```

```
i := 2;
j := 3;
P(i, j);
```

If both parameters to P are passed by reference, what are the values of i and j at the end of the program fragment?

A. $i = 0, j = 2$
B. $i = 1, j = 5$
C. $i = 2, j = 3$
D. $i = 3, j = 2$
E. None of the above

Solution 7.
When parameters, to functions or procedures, pass by reference or pointer (also referred to as by name), the changes made in the function or procedure are reflected in the caller. When parameters pass by value, the change is not reflected. Thus when i *and* j go with the values 2 and 3 respectively, k *and* m will get these values in that order and then k will have the value 2 - 3 = -1, m will have a value -1 + 3 = 2, and finally k will have a value 2 - -1 = 3. These values are sent back to i *and* j resulting $i = 3$ *and* $j = 2$. Hence the answer is D.

**Observe that the procedure swaps the values of i *and* j with out using any extra memory. BUT it is very wrong to use such a function in applications which deal with big absolute value numbers. As such suppose that i is assigned the largest positive integer value possible of the programming language under consideration and let j be assigned any negative integer, then the first line in the function will assign k a value above the integer limit resulting to a wrong result!

Question 8.
A starvation-free job-scheduling policy guarantees that no job waits indefinitely for service. Which of the following job-scheduling policies is starvation-free?

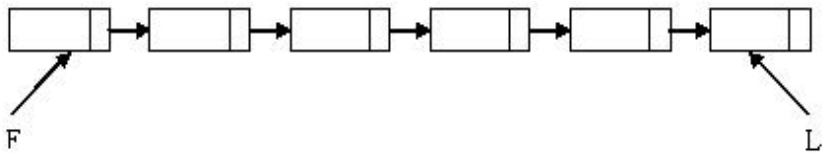
A. Round-robin
B. Priority queuing
C. Shortest job first
D. Youngest job first
E. None of the above

Solution 8.
Consider Priority queuing; a policy where the process with the highest priority is executed first irrespective of its arrival. Under such policy, suppose a system is currently executing a process and is in the middle of execution when a process with higher priority arrives. This means the system will suspend the execution of the process which was already executing and will start executing this new process. If this new process needs some resources which are already occupied by the former process then a deadlock occurs as neither the higher priority process will complete execution

without these resources nor the old process can proceed executing in order to release resources for the new process. Thus there can be a deadlock (≡starvation). The same argument applies for shortest job first policy in which case a shortest process might need resources that are occupied by longer processes which were already in the middle of their execution. Youngest process first policy is where the process that arrived last will execute first which again can have starvation. The Round-robin policy which is preemptive scheduling algorithm forces a process if it seems it is holding resource which are required by others thus avoiding starvation. Thus the answer is A.

Question 9.

Consider a singly linked list of the form where F is a pointer to the first element in



the linked list and L is a pointer to the last element in the list. The time of which of the following operations depends on the length of the list?

A. Delete the last element of the list
B. Delete the first element of the list
C. Add an element after the last element of the list
D. Add an element before the first element of the list
E. Interchange the first two elements of the list

Solution 9.

Remember that after performing any operation, the structure of the list must remain intact; in other words F and L must point to the first and last elements respectively.

Choice B needs only the operation F = F->next;

Choice C needs only the operations L->next = new node, L = new node;

Choice D needs only the operations new node->next = F, F = new node;

Choice E needs only the operations T=F, F=F->next, T->next=F->next, F->next=T;

All these do not depend on the length of the list. The answer is therefore A. Indeed in order to delete the last element from the list, we need to first locate the element before the last (which can not be accessed from L). Thus we must parse all the list from the first till the element just before the last after which we can delete the last element and assign L to the one before.

Question 10.

```
p := 1; k := 0;
while k < n do
begin
    p := 2 * p;
```

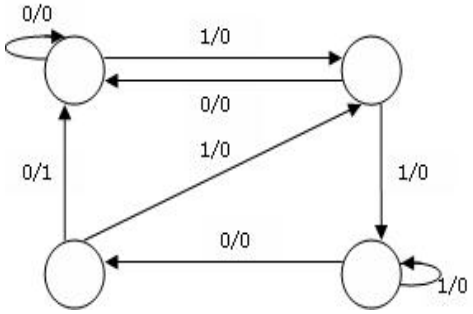
```
        k := k + 1;
    end;
```

For the program fragment above involving integers p, k, and n, which of the following is a loop invariant; i.e., true at the beginning of each execution of the loop and at the completion of the loop?

A. $p = k + 1$
B. $p = (k + 1)^2$
C. $p = (k + 1)2^k$
D. $p = 2^k$
E. $p = 2^{k+1}$

Solution 10.
Choice E is incorrect even before the loop started hence it is out. When one loop is done, the value of p is 2 and that of k is 1. At this time, we see that both choices B and C are incorrect and thus they are out. Performing one more loop gives p a value 4 and k will have a value 2 at which time choice A becomes incorrect. Hence the answer is choice D.

Question 11.
Consider an output-producing, deterministic finite automaton (DFA) of the kind indicated in the figure below, in which it is assumed that every state is a final state. Assume that the input is at least four bits long. Which of the following is (are)



true?
I. The last bit of the output depends on the start state.
II. If the input ends with "1100", then the output must end with "1".
III. The output can not end with "1" unless the input ends with "1100".

A. I only
B. II only
C. I and II only
D. II and III only
E. I, II, and III

Solution 11.

We start our analysis of the DFA by points II and III. First of all note that a label a/b on an edge denotes that with an input a, we do transition along the edge and produce an output b. Now, if the input ends with "1100", then we can verify, by starting from any of the states, that the last output bit is always a '1'. Hence point II is true. To analyze III we proceed as follows. Denote the states (circles in the diagram) by $A, B, C,$ *and* D starting from the bottom left and going in the clockwise direction. In order to get an output '1' as the last output bit, the last transition must be from state A to state B which is equivalent to say the last input bit was a '0'. We can get to state A only from state D which means the second from the last output bit is a '0' and the second from the last input was a '0' too. To get to state D, either we must have come from state D itself or from state C. This means, in any case the third from the last output was a '0' and the third from the last input was a '1'. If we had come to state D from itself, then the fourth, from the last, output bit was a '0' and the fourth, from the last, input bit was a '1'; if on the other hand we had come from state C then we must had come to state C either from state A or from state B in which in any case gives the fourth, from the last, output bit was a '0' and the fourth from the last input bit as a '1'. Thus we conclude that in order to get a '1' as the last bit of the output, we must have the last four bits of the input to be "1100". Hence III is true. By II and III, we conclude that I is incorrect. Hence the answer is choice D.

Question 12.

A particular BNF definition for a "word" is given by the following rules.

```
<word> ::= <letter> | <letter><pairlet> | <letter><pairdig>
<pairlet> ::= <letter><letter> | <pairlet><letter><letter>
<pairdig> ::= <digit><digit> | <pairdig><digit><digit>
<letter> ::= a | b | c | ... | y | | z
<digit> ::= 0 | 1 | 2 | ... | 9
```

Which of the following lexical entries can be derived from $< word >$?
I. word
II. words
III. c22

A. None
B. I and II only
C. I and III only
D. II and III only
E. I, II, and III

Solution 12.

Observe that both $< letter >$ and $< digit >$ produce only a single character, both $< pairlet >$ and $< pairdig >$ produce even number of characters; hence $< word >$ produces odd number of characters. Thus I is automatically incorrect. II can be produced with the following chain of rules:

```
<word> = <letter><pairlet>
```

```
       = <letter><pairlet><letter><letter>
       = <letter><letter><letter><letter><letter>
       = words
```

and III can be produced with the following chain of rules:

```
<word> = <letter><pairdig>
       = <letter><digit><digit>
       = c22
```

Hence the answer is choice D.

Question 13.
Consider the following C-like program.

```
#include <stdio.h>
main()
{
    float sum = 0.0, j = 1.0, i = 2.0;
    while (i/j > 0.001)
    {
        j = j + j;
        sum = sum + i/j;
        printf("%f\n", sum);
    }
}
```

How many lines of output does the program produce?

A. 0 - 9
B. 10 - 19
C. 20 - 29
D. 30 - 39
E. More than 39

Solution 13.
Since there is one line of output for each loop, we need to determine the number of times the loop executes. Since i is constant, we need to see the growth of j only. Let the initial value of j be denoted by J_1 and the subsequent values by J_n for n= 2, 3, ... so that J denotes a progression of j. We see that $J_n = 2J_{n-1}$; $J_1 = 2$, which gives $J_n = 2^{n-1}$, $n = 1, 2, ...$ The loop will execute as long as $i/j = 2.0/2^{n-1} > 0.001$ which gives $n < 3log_2 10 + 2$ or $n < 11.97$ Thus the loop will execute 11 times which is equivalent to say there will be 11 lines of output. Hence the answer is choice B.

Question 14.
For the C-like code shown above, which of the following is the integer that best approximates the last number printed?

A. 0

B. 1
C. 2
D. 3
E. 4

Solution 14.
We need the value of sum at the last loop execution which is $0 + 1 + \frac{1}{2} + ... + \frac{2}{2048}$ = 1.999 Thus the answer is choice C.

Question 15.
An integer c is a common divisor of two integers x and y if and only if c is a divisor of x and c is a divisor of y. Which of the following sets of integers could possibly be the set of all common divisors of two integers?

A. $\{-6, -2, -1, 1, 2, 6\}$
B. $\{-6, -2, -1, 0, 1, 2, 6\}$
C. $\{-6, -3, -2, -1, 1, 2, 3, 6\}$
D. $\{-6, -3, -2, -1, 0, 1, 2, 3, 6\}$
E. $\{-6, -4, -3, -2, -1, 1, 2, 3, 4, 6\}$

Solution 15.
Since 0 can not divide any number, we see that choices B and D are out of question. If an integer a is among common divisors of two integers, then surely all divisors of a must also be in the set of common divisors; thus choice A is incorrect since 3 must also be in the set. If both 3 and 4 are among common divisors, then so should be 3 * 4 = 12 by Euclid's lemma ($\because$ if we let the two integers x and y, then since $4|x$, we must have $x = 4q$ for some integer q and thus $3|4q$ with gcf(3,4)= 1 $\implies$ $3|q \implies q = 3p$ for some integer p and thus $x = 4q = 4*3*p = 12p \implies 12|x$. Similar argument for y.) Thus choice E is incorrect and the correct answer is choice C.

Question 16.
Consider the following grammar.

```
S ::= AB
A ::= a
A ::= BaB
B ::= bbA
```

Which of the following is false?

A. The length of every string produced by the grammar is even.
B. No string produced by the grammar has an odd number of consecutive b's
C. No string produced by the grammar has three consecutive a's
D. No string produced by the grammar has four consecutive b's
E. Every string produced by the grammar has at least as many b's as a's

Solution 16.

Observe that the length of any substring produced by B is 2^n plus length of substring produced by A for some integer n greater than or equal to 1. Hence the length of substring produced by B is odd if that of A is odd and is even if otherwise. This means both A and B produce same type (either both odd or even) lengths. This implies the length of string produced by S is always even. Alternatively one may reason as follows. The length of substring produced by A is either 1 (which is odd) or is 1 plus twice of the length of B (which is odd again). Hence A produces odd length substrings. B produces substrings of length 2 plus length of substrings produced by A (which is odd). Therefore B produces odd length substrings. Hence S produces odd + odd = even length strings. Thus choice A is true. As to for choice B, it is easy to see that 'b' is always produces in consecutive pairs. Hence choice B is true. Choice C is also true after all 'A' and 'a' can be next to each other only once which gives a maximum of two consecutive a's. The construction S = AB = BaBB = BabbAB = BabbBaBB = BabbbbAaBB already shows that 4 consecutive b's is possible. Hence choice D is False. It is easy to see that B gives two b's and an A which itself gives only an 'a' or as many as b's as a's. Hence choice E is also true. Thus the answer is choice D.

Question 17.

A particular parallel program computation requires 100 seconds when executed on a single processor. If 40 percent of this computation is "inherently sequential" (i.e., will not benefit from additional processors), then the theoretically best possible elapsed times for this program running with 2 and 4 processors, respectively, are

A. 20 and 10 seconds

B. 30 and 15 seconds

C. 50 and 25 seconds

D. 70 and 55 seconds

E. 80 and 70 seconds

Solution 17.

The computation requires 100 seconds on single processor implies that 40% of the computation takes 40 seconds on any number of processors and the remaining 60% takes $\frac{60}{\#processors}$ seconds on parallel computation which becomes 30 seconds on two processors and 15 seconds on four. Hence, in total, the computation takes 40 + 30 = 70 seconds on two processors and 40 + 15 = 55 seconds on four processors. Thus the answer is choice D.

Question 18.

The lookup page table shown below is for a job in a page virtual storage system

Virtual Page	Actual Page
0	3
1	-
2	4
3	0

with a page size of 1024 locations. Each virtual address is in the form $[p, d]$ where p and d are the page number and the displacement in that page, respectively. A virtual address of [0, 514] maps to an actual address of

A. 514

B. 1024

C. 3586

D. 4514

E. none of the above

Solution 18.

A virtual page of 0 refers to an actual page of 3. Hence the required actual address is 3 * page size + displacement on that page = 3 * 1024 + 514 = 3586. Hence the answer is choice C.

Question 19.

If the expression ((2 + 3) * 4 + 5 * (6 + 7) * 8) + 9 is evaluated with * having precedence over +, then the value obtained is the same as the value of which of the following prefix expressions?

A. + + * + 2 3 4 * * 5 + 6 7 8 9

B. + * + + 2 3 4 * * 5 + 6 7 8 9

C. * + + 2 3 4 * * 5 + + 6 7 8 9

D. * + + + 2 3 4 * * 5 + 6 7 8 9

E. + * + * 2 3 4 + + 5 * 6 7 8 9

Solution 19.

The given expression evaluates to (6 * 4 + 5 * 13 * 8) + 9 = 553

Choice A evaluates to

+ + * 6 4 * * 5 13 8 9 = + + 24 * 65 8 9 = + + 24 520 9 = + 544 9 = 553

Choice B evaluates to

+ * + 6 4 * * 5 13 8 9 = + * 10 * 65 8 9 = + * 10 520 9 = + 5200 9 = 5209

Choice C evaluates to

* + 5 4 * * 5 + 13 8 9 = * 9 * * 5 21 9 = * 9 * 105 9 = * 9 945 = 8505

Choice C evaluates to

* + + 5 4 * * 5 13 8 9 = * + 9 * 65 8 9 = * + 9 520 9 = * 529 9 = 4761

Choice C evaluates to

+ * + 6 4 + + 5 42 8 9 = + * 10 + 47 8 9 = + * 10 55 9 = + 550 9 = 559

We conclude that the only expression that evaluates to the same result as the given expression is that in choice A. Hence the answer is choice A.

Question 20.

Let P be a procedure that for some inputs calls itself (i.e., is recursive). If P is guaranteed to terminate, which of the following statements must be true?

I. P has a local variable.

II. P has an execution path where it does not call itself.

III. P either refers to a global variable or has at least one parameter.

A. I only

B. II only

C. I and II only

D. II and III only

E. I, II, and III

Solution 20.

A recursive procedure may or may not have a local variable. Hence I is incorrect. All recursive procedures have an execution path where they do not call themselves. Hence II is correct. A recursive procedure terminates only its parameter or a global variable is being tested. Hence III is also true. Thus the answer is choice D.

Question 21.

Consider a computer design in which multiple processors, each with a private cache memory, share global memory using a single bus. This bus is the critical system resource.

Each processor can execute one instruction every 500 nanoseconds as long as

memory references are satisfied by its local cache. When a cache miss occurs, the processor is delayed for an additional 2000 nanoseconds. During half of this additional delay, the bus is dedicated to serving the cache miss. During the other half, the processor cannot continue, but the bus is free to service requests from other processors. On average, each instruction requires 2 memory references. On average, cache misses on 1 percent of references.

What proportion of the capacity of the bus would a single processor consume, ignoring delays due to competition from other processors?

A. $\frac{1}{50}$
B. $\frac{1}{27}$
C. $\frac{1}{25}$
D. $\frac{2}{27}$
E. $\frac{1}{5}$

Solution 21.
Consider any one processor as it does 100 memory references which is equivalent to say as it executes 50 instructions. Since the cache memory occurs on 1% of references, we see that on average there will be 1 cache miss on these 100 references (50 instructions executions) we are looking at. Thus these 50 instructions will take a total time of (49 * 500) ns + (1 * 500ns + 2000ns) which is (24500 + 2500)ns = 27000ns. Also since there is only one cache miss during this entire time, the processor uses the bus for only 1000ns (half of the 2000ns). Thus the required proportion is $\frac{1000ns}{27000ns} = \frac{1}{27}$. Hence the answer is choice B.

Question 22.
In multiprogrammed systems it is advantageous if some programs such as editors and compilers can be shared by several users.
Which of the following must be true of multiprogrammed systems in order that a single copy of a program can be shared by several users?

I. The program is a macro.
II. The program is recursive.
III. The program is reentrant.

A. I only
B. II only
C. III only

D. II and III only

E. I, II and III

Solution 22.

Instead of directly answering the question, let me give a very clear definition of the term reentrant (source: searchVB.com)

Reentrant is an adjective that describes a computer program or routine that is written so that the same copy in memory can be shared by multiple users. Reentrant code is commonly required in operating systems and in applications intended to be shared in multi-use systems. A programmer writes a reentrant program by making sure that no instructions modify the contents of variable values in other instructions within the program. Each time the program is entered for a user, a data area is obtained in which to keep all the variable values for that user. The data area is in another part of memory from the program itself. When the program is interrupted to give another user a turn to use the program, information about the data area associated with that user is saved. When the interrupted user of the program is once again given control of the program, information in the saved data area is recovered and the program can be reentered without concern that the previous user has changed some instruction within the program.

Of course, recursive programs are routines that call themselves from within themselves and hence have nothing to do with this question. Macro is a standard procedure except that it is ready for deployment in a special way and hence has nothing to do with multi programming. Thus the answer is choice C.

Question 23.

A particular disk unit uses a bit string to record the occupancy or vacancy of its tracks, with 0 denoting vacant and 1 denoting occupied. A 32-bit segment of this string has the hexadecimal value $D4FE2003$. The percentage of occupied tracks for the corresponding part of the disk, to the nearest percent, is

A. 12

B. 25

C. 38

D. 44

E. 62

Solution 23.

D4FE2003 in hex = 1101 0100 1111 1110 0010 0000 0000 0011 in binary. The number of 1's is 14 and the total number of bits is 32. Hence the required percentage is

$\frac{14}{32} = 43.75\%$. Hence the answer is choice D.

Question 24.

A program that checks spelling works in the following way. A hash table has been defined in which each entry is a Boolean variable initialized to $false$. A hash function has been applied to each word in the dictionary, and the appropriate entry in the hash table has been set to $true$. To check the spelling in a document, the hash function is applied to every word in the document, and the appropriate entry in the table is examined. Which of the following is (are) correct?

I. $true$ means the word was int the dictionary.

II. $false$ means the word was not in the dictionary.

III. Hash table size should increase with document size.

A. I only

B. II only

C. I and II only

D. II and III only

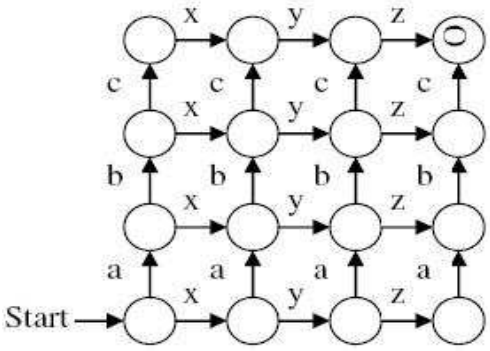
E. I, II and III

Solution 24.

We see that Hash table size has nothing to do with a document size and hence III is wrong. Thus choices D and E are incorrect. II is right because we find false if the word being checked was not in the dictionary. I is really confusing as far as I am concerned. On one hand, one may argue that it is correct after all we get true only if we naively think that the word was in the dictionary. BUT think of what happens if we test a word that does not exist in the table, let alone in the dictionary. Will the function return false or true? In my opinion this depends on the implementation. If one initializes the returned value to true, upon testing the word and not finding it on the table nothing will be done and true will be returned. In this case true does not necessarily imply that the word was in the dictionary. Hence the only correct part is II. Observe that false is obtained if the word is not in the dictionary as depicted in II or if the word is not in the table all in all which again means the word is not in the dictionary. Hence in any case II is correct. Thus the answer is choice B.

Question 25.

The finite automation below recognizes a set of strings of length 6. What is the total number of strings in the set?



A. 18

B. 20

C. 30

D. 32

E. None of the above

Solution 25.

This is a little bit difficult problem. Let x_{ij} denote the number of possible paths from a given state of the automaton to another state such that there are $i - rows$ and $j - columns$ between these two states. We see that the number of possible paths from the start state of the automaton to the final state is x_{44}. This is equal to $2x_{33} + x_{24} + x_{42} = 2(2x_{22} + x_{13} + x_{31}) + (x_{14} + x_{23}) + (x_{32} + x_{41}) = 2(2*2+1+1)+(1+(x_{13}+x_{22}))+(3+1) = 12+(1+(1+2))+4 = 12+4+4 = 20$.

Alternatively one can reason in a lengthy but more understandable way as follows. $x_{44} = x_{34} + x_{43} = 2x_{34}$ by symmetry. $x_{34} = x_{24} + x_{33} = (x_{14} + x_{23}) + (x_{23} + x_{32}) = 1+3x_{23}$ by symmetry again. $x_{23} = x_{13}+x_{22} = 1+(x_{12}+x_{21}) = 1+1+1 = 3$. Thus $x_{34} = 1+3*3 = 10$ and this implies $x_{44} = 2*10 = 20$. Again the same result. Hence the answer is choice B.

Question 26.

Let S be the statement: for $i := 1$ to N do $V[i] := V[i] + 1$
Which of the following perform(s) the same changes to V as S?

```
I.  i := 0 ;
    while i < = N do
    begin i := i + 1 ; V [i] := V [i] + 1 end

II. i := 1 ;
    while i < N do
    begin V [i] := V [i] + 1 ; i := i + 1 end
```

```
III. i := 0 ;
     while i < N do
     begin V [i + 1] := V [i + 1] + 1 ; i := i + 1 end
```

A. I only
B. II only
C. III only
D. II and III only
E. I, II and III

Solution 26.
The given code executes the loop N times. I executes N + 1 times, II executes N - 1 times while III executes N times. Hence the answer is choice C.

Question 27.

```
var i , j , x : integer ;
read (x) ;
i := 1 ; j := 1 ;
while i < 10 do
begin
    j := j * i ;
    i := i + 1 ;
    if i = x then exit
end
```

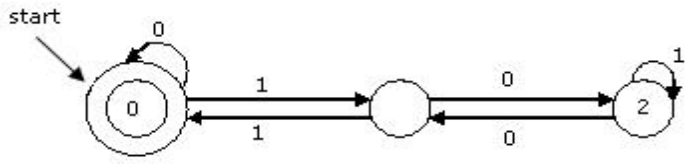
For the program fragment above, which of the following statements about the variables i and j must be true after execution of the fragment?

A. $(j = (x-1)!) \wedge (i \geq x)$
B. $(j = 9!) \wedge (i = 10)$
C. $(j = 10!) \wedge (i = 10)$
D. $((j = 10!) \wedge (i = 10)) \vee ((j = (x-1)!) \wedge (i = x))$
E. $((j = 9!) \wedge (i \geq 10)) \vee ((j = (x-1)!) \wedge (i = x))$

Solution 27.
The loop terminates when either $i = x < 10$ and $j = 1 * 1 * 2 * 3 * ... * (x-1) = (x-1)!$ OR $i = 10 \leq x$ and $j = 1 * 1 * 2 * 3 * ... * 9 = 9!$ Thus the answer is choice E.

Question 28.

State 0 is both the starting state and the accepting state



Each of the following is a regular expression that denotes a subset of the language recognized by the automaton above EXCEPT

A. $0^*(11)^*0^*$
B. $0^*1(10^*1)^*1$
C. $0^*1(10^*1)^*10^*$
D. $0^*1(10^*1)0(100)^*$
E. $(0^*1(10^*1)^*10^* + 0^*)^*$

Solution 28.
Tracing choice D we see that 0^* will take us to state 0, then 1 will take us to state 1, then 10^*1 will make us stay in the same state, the following 0 will take us to state 2, and finally $(100)^*$ will not change the state we are in. Hence we will end up in state 2 which is not an accepting state. All the other choices A, B, C and E can be traced to yield subset of the language recognized by the automaton. Thus the answer is choice D.

Question 29. *[Questions 29 and 30 are based on the code fragment below]*

```
    Program Main (input, output);
    Type
        Link = $\uparrow$ Cell;
        Cell = record
                 Info : integer;
                 Next : Link
               End;
```

If the rest of the program is as follows,

```
    var
        p1, p2 : Link;
    procedure A;
        var
            p3 : Link;
        begin
(ii)        new (p2);
(iii)       new (p3);
(iv)        new (p3$\uparrow$.Next)
        end ;
    begin (Main)
```

```
(i)     new (p1);
        A;
(v)     ...
    end
```

then, at line (v), which of the following allocated Cells is (are) available for reclamation by the garbage collector?

A. None
B. Only the one allocated by (iii)
C. Only the ones allocated by (iii) and (iv)
D. Only the ones allocated by (ii), (iii), and (iv)
E. Those allocated by (i), (ii), (iii), and (iv)

Solution 29.
When the execution reaches line (v), all LOCAL memories allocated inside the procedure A are available for reclamation. These are p_3 and $p_3 \longrightarrow next$ which were allocated memories by lines (iii) and (iv). Hence the answer is choice C.

Question 30.
In another program written in a Pascal-like language with the same heading and type declaration, assume that a choice must be made between the following two options.

```
Option P: the procedure Push defined by

        Procedure Push (n : integer; h : Link);
        Var
            p : Link;
        Begin
            new(p) ;
            p$\uparrow$.Info := n ;
            p$\uparrow$.Next := h ;
            h := p
        end;
```

To be used in the main program using a procedure call of the form:

Push (n, Head);

```
Option F: the function Push defined by

        Function Push (n : integer; h : Link) : Link;
        var
            p : Link ;
        begin
            new(p) ;
            p$\uparrow$.Info := n;
```

```
        p$\uparrow$.Next := h ;
        Push := p
    end;
```

To be used in the main program using an assignment of the form

Head := Push (n, Head);

Suppose $Head$ points to the first cell of the list of cells, and a new cell is to be added at the front of the list, leaving $Head$ pointing to the newly added cell.

Which of the following implementations will FAIL to accomplish this objective?

A. Option P with parameters passed by value
B. Option F with parameters passed by value
C. Option P with parameters passed by reference
D. Option F with parameters passed by reference
E. Option P with parameters passed by name

Solution 30.
It is easy to see that Option P must be able to change h globally and thus needs its parameter h to come by reference or pointer or name BUT not by value. Hence the answer is choice A.

Question 31.
An XY flip-flop operates an indicated by the following table.

Inputs		Current State	Next State
X	Y		
0	0	Q	1
0	1	Q	Not(Q)
1	1	Q	0
1	0	Q	Q

Which of the following expresses the next state in terms of the X and Y inputs and the current state Q?

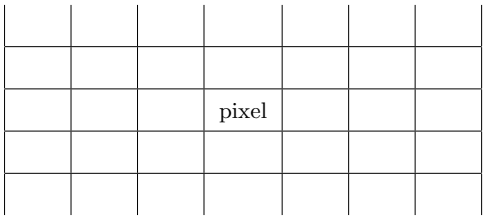
A. $(\bar{X} \wedge \bar{Q}) \vee (\bar{Y} \wedge Q)$
B. $(\bar{X} \wedge Q) \vee (\bar{Y} \wedge \bar{Q})$
C. $(X \wedge \bar{Q}) \vee (Y \wedge Q)$
D. $(X \wedge \bar{Q}) \vee (\bar{Y} \wedge Q)$
E. $(X \wedge \bar{Q}) \vee (\bar{Y} \wedge \bar{Q})$

Solution 31.

Tracing choice A shows that it fulfils the requirement hence the answer is choice A.

Question 32.

A black-and-white computer graphics display is divided up into an array of pixels as shown below.



Each of the pixels can take on one of the eight gray levels ranging from 0 (white) to 7 (black). In order to prevent sharp discontinuities of shade, the software system that causes pictures to be displayed enforces the rule that the gray levels of two adjacent pixels can not differ by more than two. How many of the 64 possible assignments of gray levels to two adjacent pixels satisfy this rule?

A. 24
B. 32
C. 34
D. 40
E. 64

Solution 32.

Consider two adjacent pixels and let the gray levels of these pixels be x and y. We need to find the cardinality of the set of all (x, y) such that $x, y \in \{0, 1, 2, ...7\}$ and $|x - y| \leq 2$. We see that

- If $y \in \{2, 3, 4, 5\}$ then x can take five different values ranging from y-2 to y+2.
- If $y \in \{0, 7\}$ then x can take three different values ranging from y to $|y - 2|$.
- If $y \in \{1, 6\}$ then x can take four different values. (why?)

Hence there are 4 * 5 + 2 * 3 + 2 * 4 = 34 possibilities all in all. Thus the answer is choice C.

Question 33.

A doubly linked list is declared as

```
Element = record
    Value : integer;
    Fwd, Bwd : ↑Element;
end;
```

Where Fwd and Bwd represent forward and backward links to adjacent elements of the list.

Which of the following segments of code deletes the element pointed to by X from the doubly linked list, if it is assumed that X points to neither the first nor the last element of the list?

A. $X\uparrow.Bwd\uparrow.Fwd := X\uparrow.Fwd;$
$X\uparrow.Fwd\uparrow.Bwd := X\uparrow.Bwd$

B. $X\uparrow.Bwd\uparrow.Fwd := X\uparrow.Bwd;$
$X\uparrow.Fwd\uparrow.Bwd := X\uparrow.Fwd$

C. $X\uparrow.Bwd\uparrow.Bwd := X\uparrow.Fwd;$
$X\uparrow.Fwd\uparrow.Fwd := X\uparrow.Bwd$

D. $X\uparrow.Bwd\uparrow.Bwd := X\uparrow.Bwd;$
$X\uparrow.Fwd\uparrow.Fwd := X\uparrow.Fwd$

E. $X\uparrow.Bwd := X\uparrow.Fwd;$
$X\uparrow.Fwd := X\uparrow.Bwd$

Solution 33.
Let the element just before x be y and that just after be z. In order to link y to z, we need $x\uparrow.Bwd\uparrow.Fwd := x\uparrow.Fwd$ and in order to link z to y, we need $x\uparrow.Fwd\uparrow.Bwd := x\uparrow.Bwd$. Hence the answer is A.

Question 34.
Processes $P1$ and $P2$ have a producer-consumer relationship, communicating by the use of a set of shared buffers.

```
P1 : repeat
        obtain an empty buffer
        fill it
        return a full buffer
    forever

P2 : repeat
        obtain a full buffer
        empty it
        return an empty buffer
    forever
```

Increasing the number of buffers is likely to do which of the following?

```
I. Increase the rate at which requests are satisfied (throughput)
II. Decrease the likelihood of deadlock
III. Increase the ease of achieving a correct implementation
```

A. I only

B. II only

C. III only

D. II and III only

E. I, II and III

Solution 34.
Increasing the memory size increases the rate at which requests are satisfied but can not alter the possibility of deadlock and neither does it play any role in implementation. Hence the only correct statement is I, thus the answer is A.

Question 35.
Which of the following instruction-set features is NOT generally considered an obstacle to aggressive pipelining of an integer unit?

A. Condition codes set by every instruction
B. variable-length encoding of instructions
C. Instructions requiring widely varying numbers of cycles to execute
D. Several different classes (sets) of registers
E. Instructions that read several arguments from memory

Solution 35.
Not answered!

Question 36. *[Questions 36 - 38 refer to the code below]*
Consider an implementation of an abstract type "set of integers" by a concrete type $IntSet$ defined as follows.

```
type
    IntSet = record
        Last : 0..Max
        V : array[1..Max] of integer
end;
```

The "add element" operation on an object S is implemented by storing the value of the element in $S.V[S.Last + 1]$ and incrementing $S.Last$, unless $Last = Max$, in which case an error flag is raised. Note that duplicate values may appear in an object's concrete representation but are hidden in its abstract representation.

Let A be the representation function from $IntSet$ into "set of integers" such that, for each concrete object S of type $IntSet$, $A(S)$ is the set of integers represented by S. $A(S)$ can be written as

A. $\{i \mid i \in 1..S.Last\}$
B. $\{i \mid i \in S.Last..S.Max\}$
C. $\{S.V[i] \mid i \in 1..S.Last\}$

D. $\{S.V[i] \mid i \in 1..Max\}$
E. $\{S.V[i] \mid i \in 1..S.V[S.Last]\}$

Solution 36.
The integers represented by S are the $S[V[i]]$ values as i runs on the populated indices which are 1 through $S.last$. Hence the answer is choice C.

Question 37.
An advantage of choosing this implementation of "set of integers" is that adding an element to a set is a contact time operation. Which of the following is a disadvantage of this implementation?

A. Adding elements to a very small sets could cause error flags to be raised.
B. Deleting elements from very large sets could cause error flags to be raised.
C. Determining whether a set is empty will require nonconstant time.
D. Constructing the union of two sets will require quadratic time in the size of the set being constructed.
E. Deleting an element from a set will require exponential time in the size of the set from which the element is deleted.

Solution 37.
Since duplicate values are stored in the array, we see that a small set with repeated elements can fill the array in which case adding element will cause error flags. Hence the answer is choice A.

Question 38.
The following code fragment mutates concrete $IntSet$ objects.

```
Procedure P(var S : IntSet, x : integer);
    var k : integer;
begin
    k := 1;
    while k <= S.Last do
    begin
        if S.V[k] = x then
        begin
            S.V[k] := S.V[S.Last];
            S.Last := S.Last - 1;
        end
        else
            k := k + 1;
    end
end
```

Which of the following abstract operations of "set of integers" does P implement?

A. Add x to S

B. Delete x from S

C. Intersect $\{x\}$ and S

D. Union $\{x\}$ and S

E. Make a copy of S

Solution 38.

It can be easily seen that the procedure deletes all instances of an element x from the array. Hence the answer is choice B. Note also why $else$ is needed to increment k. If the $else$ was not there *i.e.*, if k was incremented without testing for $else$ statement then problem would arise if we have $S.V[S.last]$ equal to x as it wouldn't be deleted.

Question 39.

To compute the matrix product M_1M_2, where M_12 has p rows and q columns and where M_2 has q rows and r columns, takes time proportional to pqr, and the result is a matrix of p rows and r columns. Consider the product of three matrices $N_1N_2N_3$ that have, respectively, w rows and x columns, x rows and y columns, and y rows and z columns. Under what condition will it take less time to compute the product as $(N_1N_2)N_3$ (.i.e., multiply the first two first) than to compute it as $N_1(N_2N_3)$?

A. There is no such condition; i.e., they will always take the same time

B. $\frac{1}{x} + \frac{1}{z} < \frac{1}{w} + \frac{1}{y}$

C. $x > y$

D. $\frac{1}{w} + \frac{1}{x} < \frac{1}{y} + \frac{1}{z}$

E. $w + x > y + z$

Solution 39.

The product $(N_1N_2)N_3$ takes time proportional to $wxy + wyz$. While the product $N_1(N_2N_3)$ takes time proportional to $wxz + xyz$. Thus have the former takes less time than the later, we must have $wxy + wyz < wxz + xyz$. Since all x, y, z, w are non negative, multiplying by $\frac{1}{xyzw}$, we find that $\frac{1}{z} + \frac{1}{x} < \frac{1}{y} + \frac{1}{w}$. Hence the answer is choice A.

Question 40.

Which of the following is usually NOT represented in a subroutine's activation record frame for a stack-based programming language?

A. Values of local variables

B. A heap area

C. The return address

D. Stack pointer for the calling activation record

E. Information needed to access nonlocal variables

Solution 40.

Values of local variables are sent to a stack, return address is also sent to stack, also information need to access non-local variables is sent to the stack as the stack is the only way to communicate to the external world, finally a stack pointer for the calling activation record is also needed to be kept in the stack as it is the only way to get back of the subroutine. But heap is not needed in stack based languages. Hence the answer is choice B.

Question 41.

Let A be a finite nonempty set with cardinality n. The number of subsets $S \subseteq A$ having odd cardinality is

A. n

B. $2^{\frac{n}{2}}$

C. 2^{n-1}

D. 2^n

E. not determinable except in terms of whether n is even or odd

Solution 41.

Recall that a set of n elements has a total of 2^n subsets. This number is even. Therefore, it seems reasonable to guess that half of these subsets have even cardinality and the remaining half have odd cardinality. One may also prove that indeed it the case, for $n > 0$, by using mathematical induction. For $n = 1$, we have two subsets: the empty set (even cardinality) and the set itself (odd cardinality). Assume it holds for a set of n elements. Now we deduce, that this implies the relation holds for a set of $n + 1$ elements as well. As such denote a set of $n + 1$ elements by a set of n elements union with a set of one element. The subsets of the set of $n+1$ elements are therefore all the subsets of the set of n elements plus the same subsets with one more element (the extra element). Since, by induction assumption, the set of n elements has 2^{n-1} odd cardinality subsets and 2^{n-1} even cardinality subsets, it follows that when one element is added to the subsets, all those with even cardinality will have odd cardinality and viceversa. Thus the set of $n + 1$ elements has in total $2^{n-1} + 2^{n-1}$ odd cardinality subsets. This gives 2^n subsets, proving the result. Thus the answer is choice C.

Question 42.

A certain algorithm A has been shown to have a running time $O(N^{2.5})$, where N is the size of the input. Which of the following is NOT true about algorithm A?

A. $\exists_{C_1,C_2}$ such that for all N the running time is less than $C_1N^{2.5} + C_2$ seconds

B. $\forall_N$, there may be some inputs for which the running time is less than $N^{2.4}$ seconds

C. $\forall_N$, there may be some inputs for which the running time is less than $N^{2.6}$ seconds

D. $\forall_N$, there may be some inputs for which the running time is more than $N^{2.4}$ seconds

E. $\forall_N$, there may be some inputs for which the running time is more than $N^{2.6}$ seconds

Solution 42.

Since the complexity $O(N^{2.5})$, by definition, denotes the upper bound, there is no input instance whose complexity should be above this upper bound. Hence the answer is choice E.

Question 43.

A data structure is comprised of nodes each of which has exactly two pointers to other nodes, with no null pointers. The following C program is to be used to count the number of nodes accessible from a given node. It uses a mark field, assumed to be initially zero for all nodes. There is a statement missing from this code.

```
struct test {int info, mark; struct test *p, *q;}
int nodecount(struct test *a)
{
    if (a->mark) return 0;
    return nodecount(a->p) + nodecount(a->q) + 1;
}
```

Which statement should be made to make the program work properly?

A. Add "a->mark = 1;" as the first statement.

B. Add "a->mark = 1;" after the if statement.

C. Add "a->mark = 1;" as the last statement.

D. Add "a->mark = 0;" after the if statement.

E. Add "a->mark = 0;" as the last statement.

Solution 43.

Since the topology or the structure is not given, it might be possible to come back to a node which was already visited through the pointers p and q. Hence to avoid

counting again, every visited node should be marked by setting its *mark* field to 1. This is done just before recursively calling the function. Hence the answer is choice B.

Question 44.

Of the following, which best characterizes computers that use memory-mapped I/O?

A. The computer provides special instructions for manipulating I/O ports.

B. I/O ports are placed at addresses on the bus and are accessed just like other memory locations.

C. To perform an I/O operation, it is sufficient to place the data in an address register and call the channel to perform the operation.

D. Ports are referenced only by memory-mapped instructions of the computer and are located at hardwired memory locations.

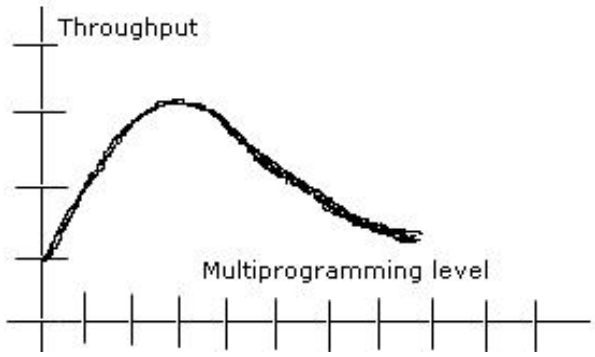
E. I/O can be performed only when memory management hardware is turned on.

Solution 44.

In memory-mapped I/O, the ports are assigned memory locations and any input/output operation is done just on those memory locations. Hence the answer is choice B. This reminds me my project which dealt specifically with this concept (`http://www.ictp.trieste.it/~weldesel`).

Question 45.

Questions 45 - 46 are based on the graph below, which shows measured values of throughput *versus* multiprogramming level for a particular computer system. ("Throughput" is defined as the rate at which requests are satisfied; "multiprogramming level" is the number of requests competing for system resources.)



At lower multiprogramming levels, throughput increases as multiprogramming level increases. This phenomenon is best explained by the fact that as multiprogramming

level increases

A. the system overhead increases

B. some system resources begins to saturate (i.e., to be utilized 100%)

C. I/O activities per request remains constant

D. the average time spent in the system by each request increases

E. the potential for concurrent activity among system resources increases

Solution 45.

At lower multiprogramming levels, the throughput increases because the potential of concurrent activity among system resources increases. Of course system overhead and saturation of resources also increase but these are not the causes for increasing throughput. Hence the answer is choice E.

Question 46.

At intermediate multiprogramming levels, the rate of increase of throughput with multiprogramming levels decreases. This phenomenon is best explained by the fact that as multiprogramming level increases

A. I/O activities per request remains constant

B. some system resources begins to saturate (i.e., to be utilized 100%)

C. the utilization of memory improves

D. the average time spent in the system by each request increases

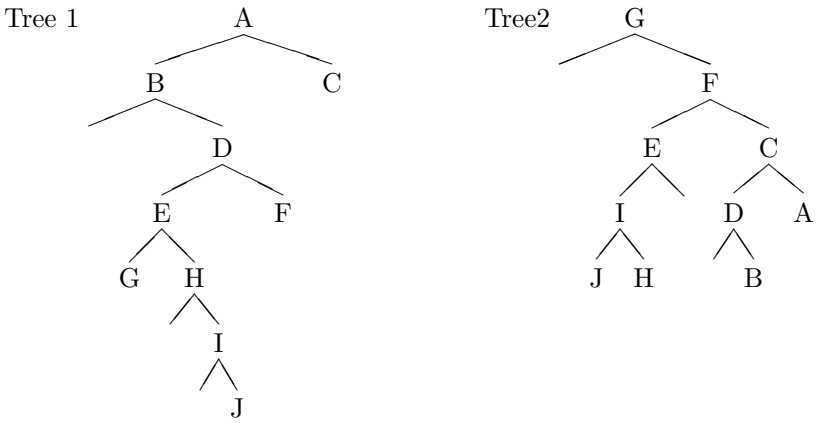
E. the potential for concurrent activity among system resources increases

Solution 46.

At higher levels of multiprogramming, some resources get saturated. But it is not necessarily true that the average time spent by requests increases. It is true for some requests but not necessarily for all. Hence the answer is choice B.

Question 47.

If Tree 1 and Tree 2 are the trees indicated below,



which traversals of Tree 1 and Tree 2, respectively, will produce the same sequence of node names?

A. Preorder, postorder

B. Postorder, inorder

C. Postorder, postorder

D. Inorder, inorder

E. Postorder, preorder

Solution 47.

The only thing to remember here is Pre-order traverses Node, Left, Right; Post-order traverses Left, Right, Node; and In-order traverses Left, Node, Right. Hence we find that the answer is choice B.

Question 48.

Let A be a finite set with m elements, and let B be a finite set with n elements. The number of distinct functions mapping A into B is

A. n^m

B. $\frac{n!}{(n-m)!}$

C. $n!$

D. $\frac{n!}{(m!(n-m)!}$

E. 2^{n+m}

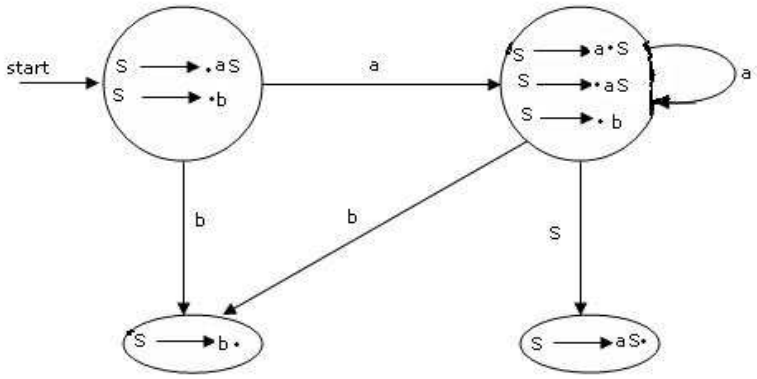
Solution 48.

Take one element from set A. It can be mapped to any of the n elements of set B. The same applies to all the elements of set A. Hence there are $n * n * n * ... * n$ mappings. This gives n^m ways. Hence the answer is choice A.

Question 49.

$$S \rightarrow aS|b$$

The "parsing automaton" below is for the context free grammar with the productions indicated above.



Each state includes certain "items", which are productions with dots in their right sides. The parser using this automaton, with X1X2...Xn on the stack, reduces by production A '? if and only if there is a path, labeled X1X2...Xn from the start state to a state that includes the item A '? . (note the dot at the right end). Which of the following stack contents causes the parser to reduce by some production?

A. a

B. aa

C. bb

D. aaS

E. ϵ

Solution 49.

Label the states, starting from top left going clockwise direction, by 1, 2, 3 and 4. Observe that the parser reduces as required only in the states 3 and 4. Now, if the stack contents are either 'a' or 'aa', then we end up in state 2 hence the parser does not reduce as required. If the stack content is 'bb' then the machine will hang as the second 'b' will create ambiguity. The empty string will end at state 1 hence no reduction again. If the stack content is 'aaS', then the first a will take the machine to state 2, the second a will keep it in state 2 and the S will take it to state 3 which includes the item "$S \rightarrow aS.''$ which is the required criteria for reduction. Thus the parser reduces as required. Hence the answer is choice D.

Question 50.

Let k $\geq$ 2. Let L be the set of strings in $\{0,1\}^*$ such that $x \in L$ if and only if the number of 0's in x is divisible by k and the number of 1's in x is odd. The minimum number of states in a deterministic finite automaton (DFA) that recognizes L is

A. $k+2$

B. $2k$

C. $klogk$

D. k^2

E. 2^k

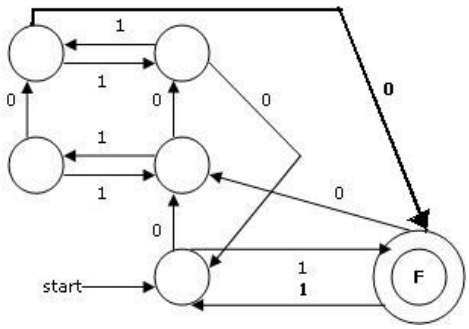
Solution 50.

I really don't know the answer to this question. I have never come across such a question. But we can try some trial and error tricks. Let's take a particular k and try to find a DFA with minimum number of states that fulfils the requirement. To that end, we may pick $k = 2$. However, problem will arise as all the choices ($k+2$, $2k$, k^2, and 2^k) will evaluate to the same value for $k = 2$. So we better kick off $k = 3$.

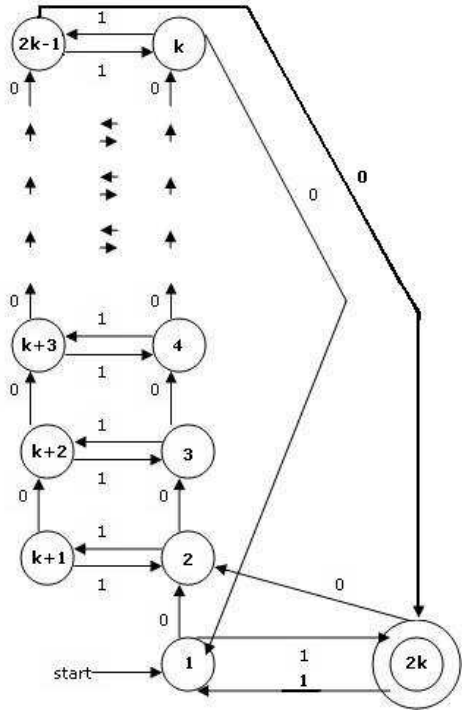
In order to see how many states we need to construct a DFA that recognizes strings over $\{0,1\}^*$ with odd number of ones and a multiple of 3 zeros, we proceed as follows: Obviously we must have a start state. Denote it by S_{Start}. With a one, we must go to a final state, hence there should be a second state which is a final state. Denote it by S_{Final}. At this step, if a one comes we may go back to the start state so as to ensure that with an odd number of ones, we can always come from the start state to the final state. Hence, our next job should be to ensure that we are at the start state if and only if we have already traced a multiple of 3 zeros. So, suppose that we encounter a zero, when we are at the start state; what should be done? Obviously we can not go to the final state; neither can we stay in the start state. Hence we should have another state, denote it by S_1, to which we should make a transition. If another zero is encountered when we are in state S_1, then we should make a transition to non of the aforementioned states. Hence there should be yet another state, denote it by S_2 to which we should make a transition. Once we are in state 2, if a zero is encountered, we can transit to the start state because we have made a multiple of 3 zeros in total after which the start state can take us to the final state with odd number of ones. So far so good with FOUR states.

But what should be done if a one is encountered when we are in state S_1? We can go to none of the states again. Hence we need to add a state, call it S_3 which will bring us back to S_1 if a one is encountered again. The same applies to state S_2 which must go to a new state, call it S_4, whenever a one is encountered and will come back to S_2 if a one is encountered again. This means we will arrive in state S_1 after $3x+1$ (for some integer x) zero and even number of ones, in state S_2 after $3x+2$ zeros and

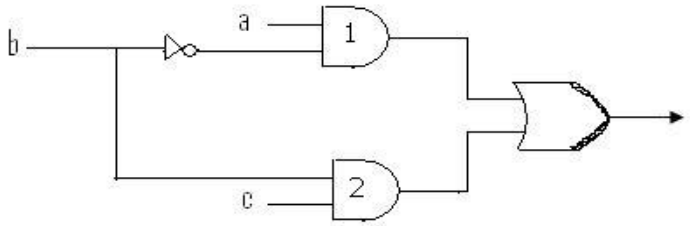
even number of ones, in state S_3 after $3x+1$, zeros and odd number of ones, and in state S_4 after $3x+2$ zeros and odd number of ones. Thus states S_3 and S_4 should bother only on number of zeros which implies state S_3 should go to state S_4 with a zero and state S_4 should go to state S_{Final} with a zero. Also the final state should transit to state S_1 if a zero is encountered. Though this construction is not enough proof, I bet it is impossible to reduce the number of states required and yet do same job. Thus the answer must be choice B. The following figure shows the DFA we just constructed. The above figure (the orientation of the states, to be precise) is found



to lead to a general construction procedure for any k. The following diagram can be inferred with no difficulty and is enough to convince the reader that the answer is in fact choice B.



Question 51.



In the diagram above, the inverter and the AND-gates labeled 1 and 2 have delays of 9, 10 and 12 nanoseconds, respectively. Wire delays are negligible. For certain values of a and c, together with a certain transition of b, a glitch (spurious output) is generated for a short time, after which the output assumes its correct value. The duration of the glitch is

A. 7ns

B. 9ns

C. 11ns

D. 13ns

E. 31ns

Solution 51.

This can easily be understood by putting a, b, and c at the same moment and see how much it takes for actual result to get to the end. The inverter will takes 9 nanoseconds to generate $\bar{b}$ and the first AND gate takes extra 10 nanoseconds to generate $a\bar{b}$. This means, it will take 19 nanoseconds for $a\bar{b}$ to get to the output. But bc will take only 12 nanoseconds to get to the output. Hence there will be a glitch for 19 - 12 = 7 nanoseconds. The answer is therefore choice A.

Question 52.

Of the following sorting algorithms, which has a running time that is at LEAST dependent on the initial ordering of the input?

A. Insertion sort

B. Quicksort

C. Merge sort

D. Selection sort

E. Shell sort

Solution 52.

Merge sort, as we all know, works by dividing the input in to two parts (blindly) and working with these two halves independently and finally merging them as its name implies. Thus it has least dependence on the initial ordering. Hence the answer is choice C.

Question 53.

A certain well-known computer family represents the exponents of its floating-point numbers as "excess-64" integers; i.e., a typical exponent $e_6e_5e_4e_3e_2e_1e_0$ represents the number $e = -64 + \sum_{i=0}^{6} 2^i e_i$. Two such exponents are input to a conventional 7-bit parallel adder. Which of the following should be accomplished in order to obtain a sum that is also in excess-64 notation?

A. The most significant adder output bit should be complemented.

B. An end-around carry should be generated.

C. The adder outputs should be bitwise complemented.

D. A low-order carry should be inserted.

E. The adder output should be left unchanged.

Solution 53.

When two exponents, each with excess 64, are added, the result will be excess 128. Hence we need to subtract (or add) 64 to the sum. Adding 64 inverts the most significant bit. Hence the answer is choice A.

Question 54.(Questions 54-55 are based on the following information)

Consider a virtual memory with M resident pages and a periodic page reference string

$p_1, p_2, ..., p_N, \quad p_1, p_2, ..., p_N, \quad p_1, p_2, ..., p_N$

of N distinct requests.

Assume that the string of N requests was constructed randomly, and assume that initially none of the pages are resident.

Assume that $N = 2M$ and FIFO is used. If the string $p_1, p_2, ..., p_N$ is repeated three times, then the number of page faults is

A. $\frac{N}{2}$

B. N

C. $N + 3$

D. $2N$

E. $3N$

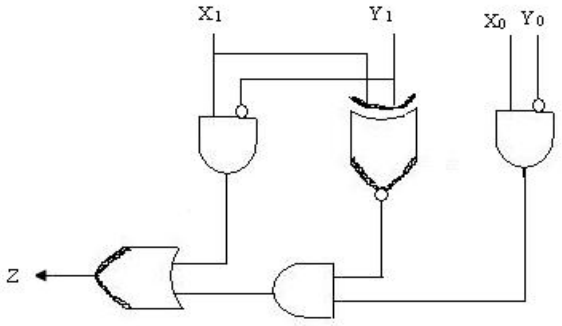
Solution 54.
Not answered!

Question 55.
If the string of N requests is repeated many times, which of the following is (are) true?
I. The fewest page faults occur when $M \geq N$.
II. LRU and FIFO will always result in the same number of page faults.
III. LIFO will always result in the fewest possible page faults.

A. I only
B. III only
C. I and II
D. I and III
E. II and III

Solution 55.
Not answered!

Question 56.



The logic circuit above is used to compare two unsigned 2-bit numbers, $X_1X_0 = X$ and $Y_1Y_0 = Y$, where X_0 and Y_0 are the least significant bits. (A small circle on any line in a logic diagram indicates logical NOT.)
Which of the following always makes the output Z have the value 1?

A. $X > Y$
B. $X < Y$

C. $X = Y$

D. $X \geq Y$

E. $X \neq Y$

Solution 56.

If $X = Y = 00$, then we get $Z = 0$. Thus choices C and D are incorrect. If $X = 00$ and $Y = 11$ then we again get $Z = 0$. Thus choices B and E are incorrect. Hence the answer is A.

Question 57.

It is known that the language $L \subseteq \{a, b\}^*$ that consists of all strings that contain an equal number of a's and b's is context-free. Let M be the regular language a^*b^*. Which of the following is (are) true?

I. $L \cap M$ is a context-free language

II. $L \cap M$ is a regular language

III. $L \cap M = \{a^n b^m | n$ is a positive integer less than integer m$\}$.

A. None

B. I only

C. III only

D. I and III

E. II and III

Solution 57.

$L \cap M = a^n b^n$. Hence III is incorrect. Also we know that $a^n b^n$ can not be recognized by DFA. Hence II is incorrect. Thus the answer is choice B.

Question 58.

Of the following, which gives the best upper bound for the value of f (N) where f is a solution to the recurrence

$f(2N + 1) = f(2N) = f(N) + logN \ for \ N \geq 1$,

with $f(1) = 0$?

A. O (log N)

B. O (N log N)

C. O (log N) + O (1)

D. O ((log N)2)

E. O (N)

Solution 58.
Not answered!

Question 59.
Let I denote the formula: $(q \implies p) \implies (p \implies q)$
Let II denote the formula $(p \implies q) \wedge q$
Which of the following is true?

A. I is not a tautology and II is not satisfiable.

B. I is not a tautology and II is satisfiable.

C. I is satisfiable and II is not satisfiable.

D. I is a tautology and II is satisfiable.

E. I is satisfiable and II is a tautology.

Solution 59.
Preparing a truth table is the simplest approach. We find that both are not tau-

p	q	p$\implies$q	q$\implies$p	I	II
T	T	T	T	T	T
T	F	F	T	F	F
F	T	T	F	T	T
F	F	T	T	T	F

tologies but are satisfiable (meaning there exist values for p and q such that I is true and same for II but possibly with different values). All the choices except B are therefore wrong. Hence the answer is choice B.

Question 60.
Consider a data type whose elements are integers and whose operations are INSERT, DELETE, and FINDCLOSEST, with FINDCLOSEST(y) defined to be some element x in the current set such that $|x-y| \leq |x_i-y|$ for all x_i in the current set. Let

$$T = max(T_{INSERT}, T_{DELETE}, T_{FINDCLOSEST})$$

where T_{op} denotes the worst-case time complexity for the given operation OP. Which of the following data structures would be best to use in order to minimize T?

A. A sorted list

B. An unordered list

C. An implicit heap

D. An AVL tree

E. A hash table with conflicts resolved by a linked list

Solution 60.

Assume sorted list is used. Then insertion will take time proportional to $logN$, deletion will take same time, and finding closest will take time proportional to N. If unordered list is used, insertion is constant time operation while both deletion and finding closest are proportional to N. Both sorted and unsorted lists therefore have $T = N$. Hence choices A and B should be incorrect. The AVL tree has $T < N$ and hence the answer is choice D.

Question 61.

A block of 105 words of memory is used for dynamic storage for objects of sizes 3 and 10 words. The operations supported by the storage are:

$x := alloc(n)$ {Allocate any block of n consecutive words and return its starting address; note that 3 and 10 are the only legal values for n.}

$free(y)$ {Make the previously allocated block starting at address y available for re-use}

At a point where a certain request for allocation of a block of words cannot be granted because of lack of a sufficiently long block of consecutive words to fill that request, what is the minimum possible number of words that might actually be in use?

A. 10

B. 24

C. 53

D. 96

E. 103

Solution 61.

The difficulty here is to understand the problem than to answer it. We need to find when can it not be possible to allocate memory when in fact there are a lot of (maximum possible) free memory words scattered all over. For this scenario, suppose that, starting from the first word, we leave nine words free and then allocate three words just above and repeat the process until we consume all the memory. Since each junk

will occupy 3 + 9 = 12 words of memory, there will be 8 such junks taking a memory of 96 words. Now suppose that a request comes looking for ten words memory. It will not satisfy; yet the actual memory in use is 8 * 3 = 24. Hence the answer is choice B.

Question 62.

Languages with a structure that implements abstract data types (e.g., c C++ class) can prevent access to components of this structure by all operations except those that are part of this structure. However, definitions of such a structure contain declarations of components of the structure (e.g., the header file for a C++ class may contain declarations of its private components).

For a such a language, an object's name could be bound at run time to stack storage for its component values (direct representation) or to a stack pointer referencing heap storage for its component values (indirect representation). Which of the following statements about comparisons between direct and indirect representations is (are) true?

I. Indirect representation noticeably increases compilation time

II. Direct representation decreases the time needed to access components of a variable.

III. When the storage size of some private component of a variable changes, indirect representation minimizes the number of recompilations of source modules that must be performed.

A. I only

B. III only

C. I and II only

D. II and III only

E. I, II, and III

Solution 62.

The use of direct representation has the advantage that accessing of components is one step process and hence is fast. Thus II is correct. Indirect representation has the advantage that the compiler need not worry about the actual storage memory once it has a pointer to the memory which makes compilation faster (decreases compilation time). Thus I is incorrect. When storage size of components changes, indirect representation helps not to recompile again those modules that have not been directly linked to the previous storage size. Hence III is correct. Thus the answer is choice D.

Question 63.(Questions 63-64 refer to the following information)

An array $A[1..n]$ is said to be $k-ordered$ if $A[i-k] \leq A[i] \leq A[i+k]$ for each i such that $k < i \leq n-k$. For example, the array 1 4 2 6 3 7 5 8 is 2-ordeered.

In a 2-ordered array of $2N$ elements, what is the maximum number of positions that an element can be from its position if the array were 1-ordered.

A. $2N-1$

B. 2

C. $\frac{N}{2}$

D. 1

E. N

Solution 63.

Consider the worst scenario in which the second element in the array is greater than the element at position $2N-1$. In this case, after sorting (1-ordered), the second element will move N positions to get into its right position. This distance is maximum achievable distance otherwise those elements greater than or equal to it (there are $N-1$ of them) will not get slots to fit in. Thus the answer is E.

Question 64.

In an array of $2N$ elements that is both 2- and 3-ordered, what is the maximum number of positions that an element can be from its position if the array were 1-ordered.

A. $2N-1$

B. 2

C. $\frac{N}{2}$

D. 1

E. N

Solution 64.

2-ordered array satisfies $a_i < a_{i+2}$ and 3-ordered array satisfies $a_i < a_{i+3}$ but this does not stop us from having $a_i > a_{i+1}$. Hence an element can be one position off its right place if it were 1-ordered. Thus the answer is choice D.

Question 65.

Consider a queue between two processors indicated below. N is the length of the queue; e, f, and b are semaphors.

```
init : e := N; f := 0; b := 1;
```

```
Processor 1:    Processor 2:
loop            loop
    P(e)            P(f)
    P(b)            P(b)
    enqueue         dequeue
    V(b)            V(b)
    V(f)            V(e)
end loop        end loop
```

Which of the following statements is (are) true?
I. The purpose of the semaphore f is to ensure that dequeue is not executed on a empty queue
II. The purpose of semaphore e is to ensure that deadlock does not occur.
III. The purpose of semaphore b is to provide mutual exclusion for queue operations.

A. I only

B. III only

C. I and III only

D. II and III only

E. I, II, and III

Solution 65.
Not answered!

Question 66.
Church's thesis equates the concept of "computable function" with those functions computable by, for example, Turing machines. Which of the following is true of Church's thesis?

A. It was first proven by Allan Turing.

B. It has not yet been proven, but finding a proof is a subject of active research.

C. It can never be proven.

D. It is now in doubt because of the advent of parallel computers.

E. It was never believed, but was assumed in order to simplify certain undecidability results.

Solution 66.
Not answered!

Question 67.
Consider the following C code.

```
int f(int x)
{
    if (x < 1) return 1;
    else return f(x-1) + g(x);
}

int g(int x)
{
    if (x < 2) return 2;
    else return f(x-1) + g(x/2);
}
```

Of the following, which best describes the growth of $f(x)$ as a function of x?

A. Logarithmic

B. Linear

C. Quadratic

D. Cubic

E. Exponential

Solution 67.

Observe that for all $x \geq 2$, we have $f(x) = f(x-1) + g(x) = f(x-1) + f(x-1) + g(\frac{x}{2}) = 2f(x-1) + g(\frac{x}{2})$. This is enough to conclude that f grows exponentially. Hence the answer is choice E.

Question 68.

Let $G = (V, E)$ be a connected, undirected graph, and let a and b be two distinct vertices in V. Let P_1 be the problem of finding a shortest simple path between a and b, and P_2 be the problem of finding a longest simple path between a and b. Which of the following statements about P_1 and P_2 is true?

A. Both P_1 and P_2 can be solved in polynomial time.

B. P_1 can be solved in polynomial time but P_2 is not known to be solvable in polynomial time.

C. P_1 is not known to be solvable in polynomial time but P_2 can be solved in polynomial time.

D. It is not known whether either P_1 or P_2 can be solved in polynomial time.

E. It is known that neither P_1 nor P_2 can be solved in polynomial time.

Solution 68.

Not answered!

Question 69.

Let T denote a nonempty binary tree in which every node either is a leaf or has two children. Then
$n(T)$ denotes the number of non-leaf nodes of T (where $n(T) = 0$ if T is a leaf),
$h(T)$ denotes the height of T (where $h(T) = 0$ if T is a leaf),
T_L denotes the left subtree of T, and T_R denotes the right subtree of T.

If F is a function defined by

$$F(T) = \begin{cases} 0 \text{ if T is leaf} \\ F(T_L) + F(T_R) + min(h(T_L), h(T_R)) \text{ otherwise} \end{cases} \quad (1)$$

the $F(T) =$

A. $n(T) + h(T) - 1$
B. $n(T) + h(T)$
C. $n(T) + h(T) + 1$
D. $n(T) - h(T) - 1$
E. $n(T) - h(T)$

Solution 69.
A simple trick answers this question completely. By the way, I learned about such tricks from 'Barrons' whilst looking for help to prepare for Toefl. So consider a tree which has just one node which is the root node. For this tree, we have $n(T) = 0, h(T) = 0, and F(T) = 0$. This makes choices A, C and D to be incorrect. Now consider a tree with three nodes (one parent node with two children). For such a tree, we have $n(T) = 1, h(T) = 1, and F(T) = 0$. This makes choice B incorrect. Hence the answer is choice E.

Question 70.
If DFA denotes "deterministic finite automata" and NDFA denotes "nondeterministic finite automata", which of the following is FALSE?

A. For any language L, if L can be recognized by a DFA, then $\bar{L}$ can be recognized by a DFA.
B. For any language L, if L can be recognized by a NDFA, then $\bar{L}$ can be recognized by a NDFA.
C. For any language L, if L is context-free, then $\bar{L}$ is context free.
D. For any language L, if L can be recognized in polynomial time, then $\bar{L}$ can be recognized in polynomial time.
E. For any language L, if L is decidable, then $\bar{L}$ is decidable.

Solution 70.

Choice A is correct because if we can recognize L using DFA, then by making few changes to the machine (just complimenting it), we can also recognize $\bar{L}$ using the same machine. The same reason applies to choice B. Choices D and E are also similarly true. Hence the only false statement is that of choice C because whilst it is true that union, concatenation and reverse of context free languages result to a context free language, intersection and complement do not necessarily give context free languages. Thus the answer is C.

!!!THE END!!!