1. Which two deficiencies prompted the design of structured languages; e.g. Pascal?

A. non-orthogonality in language primitives
B. the prevailing use of goto as a control structure
C. the lack of structured data types such as arrays
D. the use of void * in C to bypass type checking

B. because Pascal featured GOTO statements.
C. Structured programming increased stronger type checking.

Example of void pointer in C

```c
void *ptr; // ptr is declared as Void pointer

char cnum;
int inum;
float fnum;

ptr = &cnum; // ptr has address of character data
ptr = &inum; // ptr has address of integer data
ptr = &fnum; // ptr has address of float data
```

Structured Programming definition - extensive use of subroutines, and control structures. Examples include Pascal,
Features include stronger type checking, sophisticated control structures, i.e. gotos

**Othogonality** - one instruction does not property that means "Changing A does not change B".

2. What can you do in dynamically-typed JavaScript that you cannot do in C?

Select the two true statements. At runtime
A. the memory allocated to a variable can change.
B. an integer can be coerced into a floating point number.
C. a number can be coerced into a string.
D. a character can be coerced into an integer.

```
var myVar = 5; //Binding the integer 5 to variable, myVar
var myVar = "Hello!"; //
var myVar = 100.54;
var myString = "This is my string" + 5;

var hello = "Hello world!";
console.log(hello); //Outputs "Hello World"
hello += 5;
console.log(hello); //Outputs "Hello World5"
```
3. Following Chomsky's definition of grammars, if a production rule has

A. a terminal on the LHS then the language is unrestricted or context-sensitive.
B. two non-terminals on the RHS then the language is not regular.
C. both A & B

A B C

True statements of Chomsky’s definition of grammar
context-free grammars and BNF grammars are theoretically equivalent
a regular grammar is Chomsky's most restrictive form of grammar
high-level programming languages are primarily context-free languages

Proofs for A.
“UNRESTRICTED, requires only that one nonterminal appear on LHS”
“CONTEXT SENSITIVE, requires at least one nonterminal on the LHS and that the RHS contain no fewer symbols than LHS. Rules are of the form:”

4.

ATTRIBUTE GRAMMAR 4.
\( \Sigma = \{=, +, n, x\} \), where \( n \) is an integer and \( x \) is a real number

1. \(<assign> ::= <var> = <expr>\)

2. \(<expr>_1 ::= <expr>_2 + <expr>_3\)
   Predicate: \((<expr>_2.type == <expr>_3.type == \text{int}) \text{ OR } (<expr>_2.type == <expr>_3.type == \text{real})\)

3. \(<expr> ::= <var>\)
   Semantic rule: \(<var>.type \leftarrow \text{lookupType(<var>)}\)
   Semantic rule: \(<expr>.type \leftarrow <var>.type\)

4. \(<expr> ::= n\)
   Semantic rule: \(<exp>.type \leftarrow \text{int}\)
5. \( <\text{expr}> ::= x \)  
   Semantic Rule: \(<\text{expr}>.\text{type} \<- \text{real} \)

What is the purpose of Attribute Grammar 4? Select the two true statements.
A. It performs type checking for addition operations.
B. It ensures that the lvalue type matches the rvalue type in an assignment.
C. It ensures that no coercions are legal in an expression.
D. It ensures that only widening coercions are legal in an expression.

5. Consider this chunk of C code:

```c
void funA(char *Stuff) {
    Stuff[0] = 'a';
}
funA("goodbye");
```

Select the two true statements concerning the code.
A. The lvalue in the assignment statement in funA is modifiable.
B. Stuff, which is type char *, cannot be dereferenced as Stuff[0].
C. "goodbye" is type const char *.
D. The code, if it compiles, will give a segfault at runtime.

“you can't modify string literals. It's undefined behavior”

When the literal string is defined, it is created by the compiler as a constant string.

If you try to modify a constant string then the program will seg fault.

6. The goal in the following C code is to initialize a 4x5 matrix of chars with zeros.

```c
int ROW = 4, COL = 5;
char *base = &matrix[0][0];
for (i = 0; i < (ROW*COL) ; i++) {
    int row = i/COL;
    int col = i%COL;
    *(__________________) = 0;
```

Which of the following is the computation for the index into the matrix? I.e., what should (_________________) be in the above code?

A. base + (ROW*col) + col
B. base + (COL*col) + row
C. base + (COL*row) + col

7. Select the two true statements concerning C compiler coercions.
A. Assigning a signed int to an unsigned int is a narrowing coercion.
B. "int n = 7 / 2;" is a narrowing coercion.
C. "double n = 8 / 2;" is a widening coercion.
D. Assigning a short int to an int is only problematic if the short is negative.

The answer is A because data is lost from the conversion of signed to unsigned. In this conversion, you lose all the negative values when converting to unsigned. This is narrowing the range of values.

In C, 8 and 2 are int and using integer division then the result is 4. It is then widening to an int to a double.

8. Select two true statements regarding break, continue and goto statements.
A. These are all examples of unconditional and labeled control structures.
B. A break can be used in a structure other than a loop.
C. A goto in C can be used to enter a loop in the middle of the loop body.
D. C# does not support a goto.

Break can be used in a switch statement.

unconditional- goto, break, continue, redo
conditional- switch or if/else, loops, and guarded commands
unlabeled- break
labeled- goto, continue
http://docs.oracle.com/javase/tutorial/java/nutsandbolts/branch.html

9. Assume the code below is written in a dynamically-scoped language.

```plaintext
x = 99;
```
sub2 () {
    x = 55;
    sub1();
}

sub1 () {
    print(x);
}

What do you know?
A. x in sub1() is bound to 55 at runtime.
B. x in sub1() is bound to 55 by the compiler.
C. x in sub1() is bound to 99 at runtime.

A because it is dynamic scoping, meaning that it looks for the closest instance of the variable

Dynamic Scoping starts by looking for the variable down the stack until it finds the first instance of the variable.

Static Scoping will look in the scope of the function for the variable and if it is not there it will look in the .data/Global section..

Quiz 10

Study this for the quiz 10. This is question 5.

What two things do you know if you are coding in a language that uses static call frames?

Procedure foo in that language will
A. be allocated a fixed-size call frame on the runtime stack when foo is called.
B. use space in the data segment as its call frame.
C. not be able to call another procedure.
D. not be able to call itself.

(**IGNORE**)
4. Select the two true statements concerning a language that uses static chaining to implement nested subroutines.

A. The nesting depth is not known by the compiler.
B. The starting procedure for the program has a nesting depth of 1.
C. The language must be lexically scoped.
D. A reference to a non-static non-local can be bound to an address by the compiler.

8. Assume foo() is written in a lexically-scoped language that does not support nested subroutines.
```c
void foo() {
    num = 10; // an integer
}
```

Which statement is true? If the binding of num to an address was done by the
A. linker then the memory for num was allocated by the linker.
B. compiler then num's address is in the data segment.
C. both A & B

---

**LISP information**

**Keywords**

car - car() returns first in list
cdr - cdr() returns the remainder of the list
atom - single element

//use the keyword defun to define a keyword
(defun myFunction())

Sampleexam.cl

;; CMPS 350  WTR 2009  Exam 1 (Lisp Component)
;; each problem worth 5 pts                      /25
;; submit this file

;; #1.

;; Define a recursive function that returns log2(n), where n = 2^k and k is an
;; integer >= 0; e.g. (log2 32 ) returns 5 and (log2 1) returns 0.

;; #2. What does this recursive function do? (write answer in comments)

(defun mystery(lst n)
    (if (null lst)
        nil
    (if (eq n 0)
        (car lst)
    (mystery (cdr lst) (- n 1))))))
Mystery iterates through a list and returns the first item of the list when N = 0
;; #3.
;; Using 'and', 'not', and 'or' only (not 'if'), define a recursive function
;; that takes a list and returns true if the list contains atom 'a'.
;; (you do not need to check sublists)

;; #4.
;; define a recursive function that takes a non-empty list and returns the
;; last element in the list.

;; #5. Given this function
(defun p5 (x)
  (if (null x)
      0
      (+ (car x) (p5 (cdr x))))
;; Complete the trace of each call to p5(n) from the initial call (p5 '(2 4 6))
;; until the final result ( 12 ).  (write your answer in comments)
;; p5 '(2 4 6) =
Each time the function is call, line 4 grabs the first item in the list, then calls itself with the rest of the list. By doing so it sums each item in the list until it is null, i.e. 2 + 4 + 6