This is the exam given a year ago. We were using a different book then, so some of the questions may not be pertinent now. Please use this as an example of the TYPES of questions, not a comprehensive list of exactly what you are to know.

Multiple Choice (2 points each) Circle the single best answer.

1. In C++, a local variable is bound to a relative offset in the activation record at what time?
   a. language definition time
   b. language implementation time
   c. compile time
   d. link time
   e. load time
   f. execution time
2. In C++, the type integer is bound to a particular number of bits at what time?
   a. language definition time
   b. language implementation time
   c. compile time
   d. link time
   e. load time
   f. execution time
3. Consider the Pascal-like type declarations below:
   ```
   type
   Name = String;
   Address = String;
   Maiden = Name;
   var
   me, you : Name
   home, work, permanent: Address
   parent: String
   birthName: Maiden
   ```
   Which set of identifiers are name equivalent?
   a) {me, you} {home, work, permanent}
   b) {home, work, permanent, parent} and {me, you, birthName,parent}
   c) {me, you home, work, permanent, parent, birthName}
   d) None of the above
4. A list comprehension is
   a. a list created from other lists
   b. a list created by listing the member elements
   c. an infinite list
   d. a cross product of lists
   e. none of the above
5. What characteristic of Haskell allows it to work with infinite lists
   a. static type checking
   b. lazy evaluation
c. Hindley-Milner type checking
d. fully curried
e. none of the above

6. Consider the Pascal-like type declarations below:
   type
   Name = String;
   Address = String;
   Maiden = Name;
   var
   me,you : Name
   home, work, permanent: Address
   grades, courses : array[1..10] of Integer
   friends: array[1..10] of Integer
   Which set of identifiers are declaration equivalent?
   a) \{me, you\} \{home, work, permanent\} \{friends, courses, grades\}
   b) \{me, you\} \{home, work, permanent\} \{courses, grades\}
   c) \{me, you, home, work, permanent\} \{courses, grades\}
   d) None of the above

7. Which type will be inferred for the following Haskell function
   \texttt{remove y [] = []}
   \texttt{remove y (x:xs) = if y==x then remove y xs else x:(remove y xs)}
   a) \texttt{Int->[Int]->Int}
   b) \texttt{Int->[Int]->[Int]}
   c) \texttt{a->[a]->[b]}
   d) \texttt{a->[a]->[a]}
   e) None of the above

8. Which type will be inferred for the following Haskell function:
   \texttt{bigger x y = [(a,b)|a<- x, b<- y, a < b]}
   a) \texttt{[a]->[b]->[c]}
   b) \texttt{a->b->[([a,b])]}
   c) \texttt{[t]->[t]->[t]}
   d) \texttt{[t]->[t]->[(t,t)]}
   e) none of the above

9. In Haskell, an anonymous function is termed
   a. fully curried
   b. Hindley-Milner
   c. lazy function
   d. lambda expression
   e. none of the above

10. The set of tokens used in a language is termed
   a. alphabet
   b. non-terminals
c. grammar
d. productions

11. Lexical analysis is
   a. turning an input stream into a series of tokens
   b. syntax analysis
   c. parsing
   d. none of the above

12. A friend says that there is a late binding of variables to types in the language you are to use for a project. What does he/she mean by this?
   a. a variable is given a type at load time
   b. a variable is given a type at compile time
   c. a variable is given a type during run time
   d. a variable is never formally associated with a type

13. The instruction sequence used to implement a built-in operator is bound at
   a. language implementation time
   b. load time
   c. run time
   d. language design time

14. Using structural type equivalence in a language like C++ has the disadvantage of
   a. An assignment statement of two equivalent types is time consuming as each individual piece must be copied separately.
   b. Similar types intended to be the same might be the different.
   c. It takes more compile time.
   d. None of the above

15. In some languages, one can specify the upper and lower bounds of each array dimension. Thus, A[2..4][5..10] has three rows and 6 columns. If this array begins at location 100 and each element takes one location, what is the accessing formula for a row major implementation?
   a. Addr(A[i][j]) = 100 + (i-2)*3 + (j-5)
   b. Addr(A[i][j]) = 83 + i*6 + j
   c. Addr(A[i][j]) = 100 + (i-2)+ (j-5)*3
   d. Addr(A[i][j]) = 100 + 6*i + 3*j
   e. None of the above

16. In C++, array A[4][10][3] is stored column major. If this array begins at location 10 and each element takes two locations what is the accessing formula?
   a. Addr(A[i][j][k]) = 10 + 2*[i + 4*j +40*k]
   b. Addr(A[i][j][k]) = 10 + 2*[4*i + 10*j +3*k]
   c. Addr(A[i][j][k]) = 10 + 2*[30*i + 12*j +40*k]
   d. Addr(A[i][j][k]) = 10 + 2*[30*i + 3*j +k]
   e. None of the above

17. In computing the accessing formula for a three dimensional array, you compute a negative virtual origin. What does this mean?
   a) You are using column major.
b) Each element of the array takes multiple locations
c) The lower bound for at least one subscript is positive
d) None of the above

18. In computing the accessing formula for an array, you compute a negative multiplier for i. What does this mean?
   a) This cannot happen
   b) The lower bound for at least one subscript is negative
   c) You are looking at a slice of another array
   d) None of the above

19. Which two grammar classes are useful to describe the syntax of programming languages
   a) regular and context sensitive
   b) context free and recursively enumerable
   c) context free and context sensitive
   d) regular and context free

20. In reading code, you see the following procedure declaration

        void procedure doit (int myarray[][3][4])

What can be concluded about array representation?
   a. row major ordering is used
   b. column major ordering is used
   c. subscripts begin at zero
   d. an array descriptor is used
   e. none of the above

21. The (partial) grammar below would be classified as

        S -> ABC
        AB -> aAD
        AB -> bAE
        DC -> BaC
        EC -> BbC
        bB -> Bb

   a. regular
   b. context free
   c. context sensitive
   d. recursively enumerable
   e. none of the above
Short Answer

1. (8 points) For the code below, show the stack of symbol tables active at line 17. Make sure to show which symbol table procedure names exist in the symbol table.

```plaintext
1.  type Color = Red|Blue|Green|Yellow
2.  integer a,c,t;
3.  integer b[Color] = {0,0,0,0};
4.  procedure Q;
5.    begin
6.      integer c,d;
7.      call P(c);
8.      print (c);
9.      end Q;
10.  procedure P(int point);
11.    integer c=0;
12.    procedure M(int z[Color])
13.      begin
14.        print z;
15.        end
16.      begin
17.        c = b[Red];
18.        Call M(b)
19.      end P;
20.  begin (main)
21.    a=1; b=2; c=3;
22.    call Q;
23.    end main
```

2. (4 points) Show an example which illustrates the following grammar is ambiguous

```
S → SS
S → (S)
S → ()
```
3. (8 points) Create a grammar of two infix operators # and @ where # has higher precedence than @ and both are right associative.

4. (8 points) Write the Haskell code to reduce contiguous identical elements in a list to one copy.
   ```haskell
   Typeit> strip [1,2,3,3,4,5,5]
   [1,2,3,4,5]
   Typeit> strip [1,2,3,4,5,2]
   [1,2,3,4,5,2]
   Typeit> strip [3,3,6,6,6,6,6,8]
   [3,6,8]
   ```

5. (8 points) Write the Haskell code to define the following function: `makePairs` which takes a non-negative integer and produces a list of pairs of integers as demonstrated below.
   ```haskell
   example,
   makePairs 7
   [(0,1), (1,2), (2,3), (3,4), (4,5), (5,6), (6,7)]
   ```

6. (4 points) Given the regular expression \( (ab)^* (b|a) \) use Thompson’s construction to produce a NFA.
7) (8 points) Given the non-deterministic automata below, use the **subset construction** to create a DFA. Show your work.

![Non-deterministic Automata Diagram]

8) (5 points) What are the advantages of lazy evaluation?

9) (5 points) What are the primary purposes of a call stack?