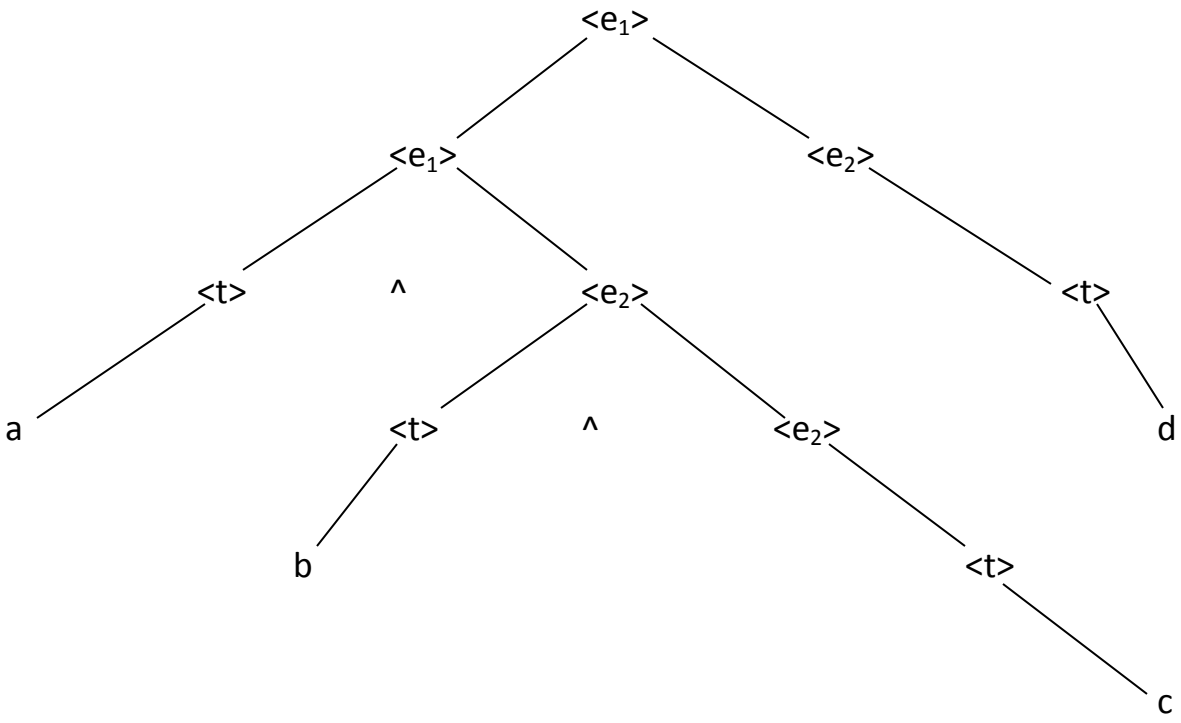
CptS 580 Advanced Programming Languages
Final Exam
Time Taken: 4 hours 45 min

Question 1:

Consider the following grammar:
$<e_1> \longrightarrow <e_2> \mid <e_1><e_2>$
$<e_2> \longrightarrow <t> \mid <t>\Lambda<e_2>$

Example:

1. a^b^cd



Question 2:

A programming language is said to have first class functions if functions are treated as first class objects (objects generated at run time). Such a language supports passing functions as arguments to other functions and returns their values to other functions assigning them to variables.

First class functions require runtime execution of the code and garbage collection technique. This leads to an increase in overhead. But there are some languages that do not need to be interpreted at runtime. Storage memory needed to store data is allocated according to the needs and is then de-allocated by a garbage collector after they are not needed anymore by the objects.

All objects are considered to be first-class data values. They are passed as arguments to procedures and function calls, return values and form new objects. Whereas in other languages, composite data values like arrays are allocated statically but is never de-allocated or is explicitly done so by the user.

Standard ML is the most functional programming language. It provides first class functions, automatic storage management (garbage collection) and static typing.

Any language that does not apply restrictions in the usage of functions and procedure calls must provide automatic garbage collection. Haskell is one of the languages which use this.

A function may create a closure and return it:

```
function derivative(f,dx) {

return function(x) {

return(f(x+dx)-f(x))/dx;

};

}
```

As the closure in this function stays for a longer time than the function that created it (scope of function), f and dx can be used even after the derivative function returns its value.

In languages without closures, lifetime of the variable coincides with the execution of scope where variable was declared. This is the implementation of garbage collection.

Question 3:

Implementation of a program which shows the order of evaluation of arguments in a function of three parameters:

1. Name of the language: C++
2. Implementation: Arguments are evaluated from right to left.
3. Source Code: File attached.
4. Output and Documentation:
   The C++ standards do not specify the order in which the arguments have to be evaluated. A given compiler can be implemented in a defined manner or an unspecified manner. Each of the compilers translates the code based on the tailoring done in the platforms.

   The order of evaluation of arguments also depends on the compiler being used.
   In the program I have executed, the arguments in the function are evaluated from Right to Left. The post-increment operation is performed on the rightmost argument first. The post increment operator is incrementing each time between the printf call statements. Hence the result obtained is a=2, b=1, c=0.

   According to C++ documentation, the order of evaluation of arguments is not specified. It is compiler dependent. The order of evaluation of arguments in various compilers is shown below:
   http://www.eetimes.com/discussion/programming-pointers/4023961/Evaluating-Function-Arguments
   "The Programming Language" By Bjarne Stroustrup, Third Edition.

Question 4:

An Iterator is an object that is used for traversals in a program by the programmer. They are implemented in terms of structures of a container implementation. Iterators perform traversals in a program and allow access to data elements in the container but do not perform iteration.
Iterators can also be implemented by using generators, which use of yield to get values to its caller several times. Tree traversal is the most common example which shows the use of iterators.

```
Iterator iter =list.iterator();
while(iter.hasNext())
{
system.out.println(iter.next());
If(iter.hasNext())
system.out.println(“, “);
}
```

To use true iterators, we must supply the loop of body as a function. They are used instead of for statements and it calls one function per iteration. The use of generators is more efficient than using true iterators. True iterator returns from the unknown function not from the function which is performing the operation.

In this problem, we need to write a Base class which allows users to create true iterators. The run() method is implemented in the IteratorHelper class and is implemented as a separate thread in the method. The state of the iterator is held in the program counter along with the local state. This cannot be done by using just a single thread. To implement the PC, we need to construct a thread. The thread created here will execute the run() part. This yields the sequential value of loop index but it should yield the value only when it is asked for, gets the next value using get() and then goes back to sleep. The yield() and get() are inter-linked. get() sends a notify signal to wake yield().The run() thread is created in the IteratorHelper.