Q1) (20pts) Regular expressions

Write regular expressions for recognizing

a) (5pts) String literals: A string literal starts and ends with a double quote. Inside the string, you could have all printable characters, except the double quote. The double quote needs to be escaped when it appears within the string. The escape character is \. In addition, the tab character and the new line character are represented as escaped characters, \t and \n, respectively. The escape character itself has to be escaped as well, as in \\. Here are some examples:

```
"abc"
"a\"bc"
"a\\bc"
"a\tbc\nde"
```

b) (5pt) Email addresses: You know what an email address looks like :D

c) (10pt) URLs: A few examples:

```
http://en.example.org/
ftp://en.example.org/
https://en.example.org:8080/
http://en.example.org/wiki/page.html
http://en.example.org/wiki/find.jsp?first_name=John&last_name=Doe
```

Q2) (20pts) Grammars and ambiguity

Consider the following grammar, where E is the set of non-terminals, and a; b; +; * are the set of terminals.

```
E → a
E → b
E → E E
E → E + E
E → E *
```

Show that the grammar is ambiguous.

Q3) (20pts) Associativity and precedence

Consider the following grammar, where `STR` is a token:

```
aexp → aexp & texp | texp
texp → uexp % texp | uexp
uexp → uexp # STR | STR
```

a) (5pts) Discuss about the associativity and relative precedence of the operators &, % and #

b) (10pts) Give a rightmost derivation for the following:

`“abc” # “def” % “uvw” % “ijk” & “klm”`

c) (5pts) Draw the parse tree.

Q4) (25pts) Left factoring and LL parsing

Consider the following grammar:

1. MapLiteral → '{' MapLiteralBody '}'
2. MapLiteralBody → Mapping ',' MapLiteralBody
3. MapLiteralBody → Mapping
4. Mapping → INT '=>' INT

a) (8pts) Left factor this grammar. Write each rule on a separate line with a unique number. I.e., do not use | to fold two rules into one. Name the new non-terminal you introduce as Shared.

b) (5pts) For each rule (there should be 5 of them), compute the FIRST() function. That is, find the terminal that can appear as the first thing on the RHS of the rule.

FIRST(1): FIRST(2): FIRST(3): FIRST(4): FIRST(5):

c) (5pts) Create an LL parse table for the new grammar.

	'{'	'}'	','	INT
MapLiteral				
MapLiteralBody				
Shared				
Mapping				

d) (7pts) Parse the following input using the parse table.

Stack	*Input*
MapLiteral	{ 3 => 5, 6 => 5 }

Q5) (15pts) Storage, scope, and bindings.

Consider the following C program.

```
int x;

void foobar()
{
  static int a;
  int b[5];
  int * c = (int *) malloc(5*sizeof(int));
  ...
}
```

For the variables x, a, b, and *c, specify the following:

a) (5pts) The scope of the variable?

x:

a:

b:

*c:

b) (5pts) Lifetime of the variable?

x:

a:

b:

*c:

c) (5pts) Storage of the variable?

x:

a:

b:

*c: