Language and Logic Assignment 2

Please submit your solutions in a pdf file in addition to your commented code and executable files with instructions on how to execute them.
Submit each file separately in uncompressed format.
The deadline for solutions is May 15

Note: Only typed solutions are acceptable. Reports that do not comply with the aforementioned requirements will not be considered.

1 Design a Turing Machine that can perform unary arithmetic without parentheses:

Your machine should support the following operations:

- Multiplication (*)
- Division (/)
- Exponentiation (ˆ)
- Modulo (%)

Notes:

1. A unary number is a number where in order to represent N, we repeat 1 N times.(11 would be 2,11111 would be 4 in decimal)
2. When the machine terminates a single unary number should be left in the tape.
3. A sample supported input would be: 11*111/11%11
4. The operator precedence is left to right, that means (((11*111)/11)%11)

What are the input alphabet and the tape alphabet?
Describe informaly how your machine works.

2 C sublanguage parser

An example of a simple sublanguage of C can be seen below:

```
struct mystruct
{
int a;
char b;
};

int main(int argc,char argv[])
{
int num=255;
int array[4];
char c='a';
if(c=='a')
{
 num=255*3;
}
int k=4;
while(k>0)
{
num=num*k+num;
}

struct mystruct astruct;
astruct.a=num;
astruct.b=c;
return 0;
}
```

The language specification is:

- characters contain any ASCII character, they are contained within ''
- a function prototype is (return type)(functionname)(parameters)
- a parameter is (type)(parametername)
- types are only integers(int), characters(char) and structs (struct (structname))
- the suported arithmetic operations are: (+,-,*,/)

- the supported boolean operations are: (|| && !)
- the supported comparison operations are (<,>,==,<=,>=)
- arrays are defined as (type)(name)[(size)] where size is a non-negative integer.
- structs members can be accessed using a . i.e. (astruct).(amember)
- variables are declared at the beginning of each block.
- functions always return.
- while and if statements are supported but not for or do.
- the function call prototype is (variable)=(functionname)(parameternames);
- for simplicity a struct can only have int or char members and not structs.

2.1 Provide a Context Free Grammar for the above language

2.2 Is your grammar ambiguous if so, remove the ambiguity

2.3 Code your own Recursive Descent parser for the above language

2.4 Use ANTLR or any other parser generator of your choice to implement the above grammar

2.5 Test your two implementations with the code below:

```
struct location
{
int x;
int y;
};

int main()
{

int snake_length=1;
location snake[10];
location food;
int u=0;
char d='L';
init_game(snake);

while(u!=1)
{
  d=handle_input();
  u=update(snake,food,d,snake_length);
  if(u==2)
  {
    food=make_new_food(snake);
    snake_length=snake_length+1;
  }
  if(snake_length==10)
  {
  u=1;
  }
}
return 0;
}

int update(location snake[],location food,char direction,int snake_length)
{
int i=0;
move_snake(snake,direction);
while(i<snake_length)
{
  if(snake[i].x==food.x && snake[i].y==food.y)
  {
  add_snake_part(snake,direction);
  }
}
return 0;
}
```

Note: you do not have to do semantic analysis to identify whether functions or variables have been declared before they are used.