Hacettepe University
Computer Science Department

Name and Surname	**: İbrahim Burak Tanrıkulu**
Identitiy Number	**: b21827852**
Course	**: BBM203**
Experiment	**: Assignment 4**
Subject	**: BNF Tree**
e-mail	**: b21827852@cs.hacettepe.edu.tr**
Main Program	**: tree.c**

2. Software Design Notes

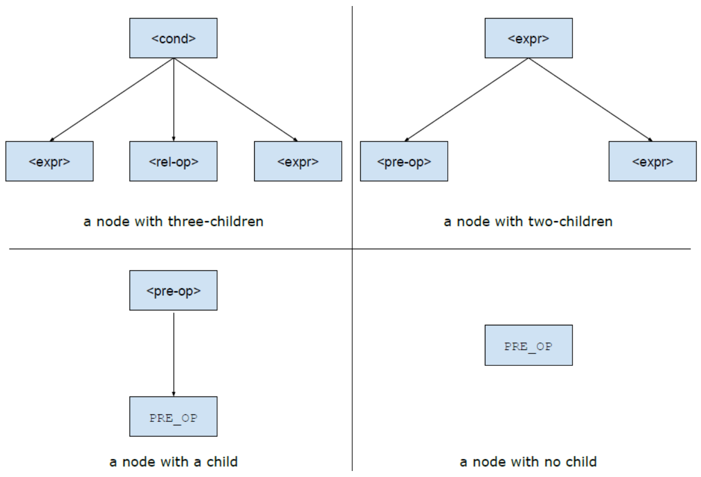
2.1. Main Data Structures

```
typedef struct {
   char* terminalSymbol;
} noChildNode;

typedef struct {
   void* firstChild;
   int childAmountOfFirstChild;
} oneChildNode;

typedef struct {
   void* firstChild;
   int childAmountOfFirstChild;
   void* secondChild;
   int childAmountOfSecondChild;
} twoChildrenNode;

typedef struct {
   void* firstChild;
   int childAmountOfFirstChild;
   void* secondChild;
   int childAmountOfSecondChild;
   void* thirdChild;
   int childAmountOfThirdChild;
} threeChildrenNode;
```

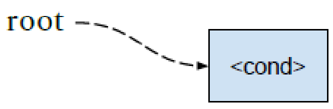


2.2. Solution

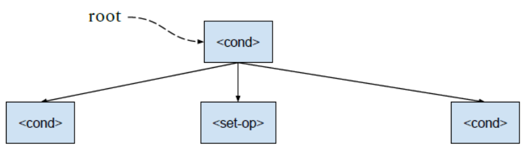
In mine data structure, i used void pointers and integers. These integers specifies children amount of child nodes. I am using these integers for casting void pointers to specific node pointers.

I created this tree recursively. I have a function that recursively creates children to given node:

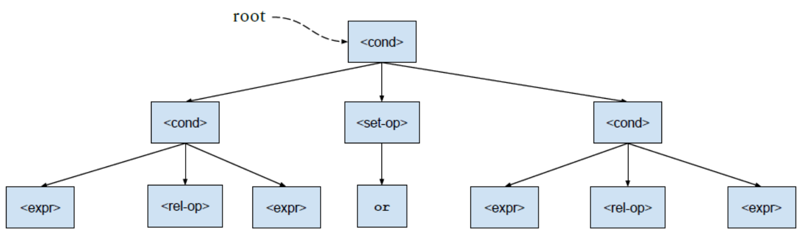
Firstly, i have a root node.



Secondly, mine function checks non-terminal symbols, selects random production rule, sets the integers and creates children to this node.



After that, the function traverses nodes and does same thing one by one recursively until reaches a terminal symbol.



We created our tree. Then, we should print it. I used another recursive function for printing this tree. Function uses integers, which are parts of the our data structure, to understand types of nodes and to cast right pointer type. This function uses in-order traverse and when encounters a terminal symbol, prints it. Else, prints paranthesis.