Birla Institute of Technology, Mesra
Department of Computer Science & Engineering
CS 333 : Compiler Design
TUTORIAL 1 (Manual Analysis); Jan 10, 2024

Consider the following C code given to you for manual analysis.

```
#include <stdio.h>
const int PRIME = 211;
int hashpjw(char* s)      // function definition
{      char* p;
       unsigned h =0, g;
       for ( p = s; *p != '\0'; p = p + 1 )   // start of loop
      {   h = ( h << 4 ) + ( *p );
          if ( g = h&0xf0000000 )   { h = h ^ ( g << 24 ) ;  h = h ^ g; }
      }        // end of loop
    return h % PRIME;
}
int main() { }
```

P1. Identify manually the tokens and lexemes that are present in the following program. Give reasonable names for the tokens (such as keyword, identifier, etc.). Present your findings in a tabular form given below. You may use the table given in the Appendix to refer to operators and their properties of C/C++ language.

Token No	Token Type	Lexeme
1	keyword	int
2	identifier	hashpjw
..	..	..

P2. Use a tuple representation for a token, (token-type, lexeme). Show the stream of tokens generated after the entire code has has been tokenized all white space (space, newline, tab) has been stripped off.

P3. Write regular expressions to specify the following lexemes in the given C code. Write a separate regular expression for each distinct token type.

- White-space
- const int for unsigned
- PRIME s p h g
- 211 0 4 0xf0000000
- = != << * ^ ( ) { }
- , ; ‘ ‘

P4. Write a regular expression for recognizing single line comments in C/C++.

APPENDIX : OPERATORS & THEIR ATTRIBUTES IN C / C++

Precedence	Associativity	Arity	Operator	Function of the operator
16	L	binary	[]	array index
15	R	unary	++, --	increment, decrement
15	R	unary	~	bitwise NOT
15	R	unary	!	logical NOT
15	R	unary	+, -	unary minus, plus
15	R	unary	*, &	dereference, address of
13	L	binary	*, /, %	multiplicative operators
12	L	binary	+, -	arithmetic operators
11	L	binary	<<, >>	bitwise shift
10	L	binary	<, <=, >, >=	relational operators
9	L	binary	==, !=	equality, inequality
8	L	binary	&	bitwise AND
7	L	binary	^	bitwise XOR
6	L	binary	|	bitwise OR
5	L	binary	&&	logical AND
4	L	binary	||	logical OR
3	L	ternary	?:	arithmetic if
2	R	binary	=, *=, /=, %=, +=, -=, <<=, >>=, &=, |=, ^=	assignment operators

End of Tutorial Sheet 1