Nalongsone Danddank Student ID : 14958950 StarID: jf3893pd

Email: [nalongsone.danddank@my.metrostate.edu](mailto:nalongsone.danddank@my.metrostate.edu)\

**Metropolitan State University**

**ICS-365-01 —Organization of Programming Languages**

**Homework #2**

1. Define *lexeme* and *token* (use your own words do not cut and paste)

Answer:

Lexemes is formal describe the syntax of programming languages. They are partitioned into some groups, For example: the name of variables, methods, classes, symbol of operator, and so forth in a programming language form a group that called “identifiers”.

Token or token of a language is a category of its lexemes because each lexeme group is represented by a token or name. For instance, an identifier is a token that can have lexemes like the name of variable (index).

1. How are programming languages formally defined.

Answer: Programming language formally defined in two distinct ways: recognition and generation.

By recognition: suppose language L that uses alphabet Σ of characters, and construct a mechanism R, recognition method that can read strings of characters from Σ of , R indicate only given input string was in L. R would accept or reject the given string. So R is a description of L.

By generation: Language generator is a device, be used to generate the sentences of a language. But it unpredictable and seem to be limited usefulness as a language descriptor.

1. Write EBNF descriptions for the following:
2. A Java class definition header statement

Answer:

<class\_head> -> {<modifier>} **class** <id> [extends class\_name]

[**implements** <interface\_name> {, <interface\_name>}]

<modifier> -> **public** | **abstract** | **final**

1. A C **switch** statement

Answer:

<switch\_stmt> -> **switch** (<expr>) {**case** <literal> : <stmt\_list> {**case** <literal>:<stmt\_list>} [**default**:<stmt\_list>]}

1. Given a context-free grammar, a recognizer for the language generated by the grammar can be algorithmically constructed? (True or False)

Answer: True

Because there is a close relationship between generation and recognition devices for a given language.

1. Using C, write a program to calculate what year someone will turn a specific age. Must use a function, and a “#defined” variable set to a target age. Say the target age is 70, your program will ask the user what their birth year is then calculate (using a function) and output the year they will be 70. Change the target age to 94, and compile and run the program again. Turn in code for and screen shots for both runs of the program.

Main function will handle all the input and output, the function (you define, not the main) will do the calculation accepting input of the birth year and returning the target year.

Answer:

#include <stdio.h>

#include <time.h>

//set target age aviable to be 70 by macro #define

#define TARGET\_AGE 70

int calculate\_age(int year);

int main(int argc, char \*argv[]) {

int year;

printf("What year is your birth?");

scanf("%d", &year);

// change target age to the correct answer.

#if defined(TARGET\_AGE)

#define TARGET\_AGE calculate\_age(year)

#endif

// then print out the correct age.

printf("Your current age is %d", TARGET\_AGE);

return 0;

}

// function for calculating from a birth year to age.

// @params year - int

// @return age - int

int calculate\_age(int year){

time\_t t = time(NULL);

struct tm tm = \*localtime(&t);

return tm.tm\_year + 1900 - year;

}

