BITS, Pilani- K K Birla Goa Campus Date: 08/05/2023

Course: CS F363 **Semester-**II, 2022-23**, Comprehensive Marks: 100** Time: 14:00-17:00

Compiler construction **Weightage:** 35% **Mode:** Closed Book Duration: 180 min

1. Each record in a subtitles file shown below has three lines: the first is the sequence number, the second shows the start and end time, in hours, minutes, seconds (separated by colons). The start and end times are separated by `-->`. The third line is the dialogue text that must appear for the time duration between the start and end times on the screen. Complete the `flex` program fragment that reads this subtitle file and adds a delay of 7 seconds to each dialog, such that each dialogue would appear and stay 7 seconds later than it would using the old subtitles file (but the durations of the staying do not change). **[10]**

Original file

```
61
00:04:52 --> 00:04:54
When I was just a child in school
62
00:59:55 --> 00:59:58
I asked my teacher What should I try
```

Output by the program should be:

```
SEQ: 61
00:04:59  --> 00:05:01
When I was just a child in school
SEQ: 62
01:00:02  --> 01:00:05
I asked my teacher What should I try
```

Complete Pattern1, 2, 3 and TODO code

```
%{
#include <time.h>
void adddelay(char *, char *, int);
%}
%%
Pattern1 {/* TODO: add proper print stmt*/}
Pattern2 {char s[100]; adddelay(yytext,s,7); printf("%s",s);}
"-->"      { printf("␣␣%s␣",yytext); }
Pattern3 { /* TODO: Print dialog text */ }
%%
void adddelay(char *buf, char *mod, int delay)
{
    struct tm result;  int SS, MM, HH;
    strptime(buf, "%T", &result); // string to time
        conversion
    SS= result.tm_sec + delay;
    // TODO: Code to complete the delay calculation
    result.tm_sec= SS; result.tm_min= MM; result.tm_hour= HH;
    strftime(mod, sizeof(mod), "%T", &result); //time to
        string conversion
}
```

2. Consider a grammar that generates a binary number with a decimal point. $1.S \rightarrow L \bullet L$ $2.S \rightarrow L$ $3.L \rightarrow L\ B$ $4.L \rightarrow B$ $5.B \rightarrow 0$ $6.B \rightarrow 1$.

   (a) Calculate the LR(0) set of items **[3]**

   (b) Create the parsing table so that there is no conflict. Then show the step by step parsing of the string `110•01` **[13]**

   (c) Create a S-attributed SDD and semantic rules to compute the decimal value of this binary number. For instance, `110•011` should be 6.375. Rule: You must use attribute *val* for all NTs to store the intermediate (as well as final) computations. Hint: You may require another attribute of a non-terminal to compute the fractional part. **[10]**

   (d) Draw an annotated parse tree showing the detailed computation of attribute values for the string `10•01`. Show the computation of the decimal value of this string at the root node of the tree. **[8]**

3. Consider the following two rules, each representing a flow-of-control construct, as in the programming language `C`. You must use the usual S and L attributes of S and C, such as $S.next$, $S.code$, $C.code$, $C.t$, $C.f$. Use the function `new Label()` to generate a new label. Assume that the last intermediate code in $C.code$ jumps either to $C.t$ or to $C.f$, depending on whether C is true or false. Use `goto L` to jump to a particular label `L`. Answer the following.

   (a) For $S \rightarrow$ `if` (C) S_1 `else` S_2, the partially complete L-attributed SDD is $\{S_1.next = S.next$, $L =$`new Label()`, $C.f =$L $\}$. Write the value of $S.code$ to complete the SDD. After completing the SDD, convert this SDD to an SDT. **[4]**

   (b) For $S \rightarrow$ `do` S_1 `while` (C), write the L-attributed SDD and the corresponding SDT. **[6]**

P.T.O. . . .

4. For $A \rightarrow BCD$, A, B, C, and D have two attributes: s: synthesized, and i: inherited. For each of the sets of rules below, tell whether (i) the rules are consistent with an S-attributed definition (ii) the rules are consistent with an L-attributed definition.
   1. $A.s = B.i + C.s$, $D.i = A.i + B.s$
   2. $A.s = B.s + D.s$
   3. $A.s = D.i$, $B.i = A.s + C.s$, $C.i = B.s$, $D.i = B.i + C.i$ $\boxed{\mathbf{6}}$

5. Answer the following **in the context of the Lab Assignment 1 through 4** :

   (a) Show which parts of function code generation are handled by the AST based SDT and which parts need more processing. Specify the data structures (classes, methods) and their behaviours for both. Here conceptual clarity is important, not syntax and exact prototypes. $\boxed{\mathbf{10}}$

   (b) Which two `LLVM` classes are `GlobalValue`s in our implementation? Why? $\boxed{\mathbf{8}}$

   (c) Specify each of `dbg` and `main` in our source language as a function, a macro, or a keyword. Justify your answers. $\boxed{\mathbf{10}}$

   (d) Choose which of the following source code snippets will be encapsulated in a single `BasicBlock` (or `BB`, in older versions) object. $\boxed{\mathbf{8}}$

#	**Source Snippet**	Remark

1	let a : int = 5;
	let b : long int = 5;	

2	let a : int = f(b);	Assume suitable `fun f` to be available.

3	fun add(a: int, b: int) {
	ret (a+b);
	}	

4	if(0) {
	dbg 1;
	}
	else {
	dbg 0;
	}	

   (e) For implementing scope rules in our language, with `if` blocks and functions, is it useful to maintain a global symbol table or a symbol table local to a basic block or local to a function? Choose and justify your option. $\boxed{\mathbf{4}}$

END