INDIAN INSTITUTE OF TECHNOLOGY, KHARAGPUR

Date 25.11.2009 ~~FN~~ / AN *Time:* ~~2~~/3 Hrs. *Full Marks* .75.... *No. of Students* ..69..
Autumn / ~~Spring~~ Semester, 2009-2010 Deptt. .CSE....... Sub No. CS31003.
3rd Yr. B. Tech.(Hons.) / ~~B. Arch.~~ / ~~M. Sc.~~ Sub. Name ...COMPILERS............

Instructions : **Answer question 1 and 2, and any one from the remaining two questions.**

1. $[8 + 5 + 8 + 5 + 4]$

Consider the following ambiguous grammar for expressions with n infix, binary operators, $@_1, @_2, \cdots, @_n$.

$$E \rightarrow E\,@_1\,E \mid E\,@_2\,E \mid \cdots \mid E\,@_n\,E \mid (E) \mid \texttt{id}$$

(a) Construct the *canonical* LR(0) collection for the grammar and express the number of states of the LR(0) automaton as a function of n, the number of binary operators.

(b) How many *shift/reduce* conflicts will be reported by the *bison* parser generator for the grammar when $n = 1$ and $n = 5$. Express the number of *shift/reduce* conflicts as a function of n.

(c) Let there be three operators $@_1, @_2, @_3$. All of them are *left associative* and the *precedence relation* is $@_1 < @_2 < @_3$. Give the complete SLR parsing table for this expression language (L_3).

(d) If for the grammar with three operators (**1c**), no information of *associativity* and *precedence* are specified in *bison*, it takes the default action to construct the parsing table. Show the differences in the table entries in this case compared to your SLR table of **1c**.

(e) Design an unambiguous expression grammar for L_3 incorporating the precedence and associativity specified in the **1c**.

2. Consider the following program. $[10 + 4 + 8 + 8]$

```
main // Selection sort
    int data(100) ;
    print "Enter the number of data:\n"
    scan int n
    print "Enter the data:\n"
    int i = 0
    while i < n {
        scan data[i] i = i + 1
    }
    i = 0
    while i < n-1 {
       int imin = i int minD = data[i]
       int j = i + 1
       while j < n {
          if data[j] < minD { imin = j minD = data[j] }
          j = j + 1
       }
       int temp = data[i] data[i] = data[imin] data[imin] = temp
       i = i + 1
    }
    i = 0
    while i < n {
        print data[i]
        i = i + 1
    }
end
```

(a) Translate the program to a sequence of *3-address codes*, identify the *basic blocks* and show the *flow graph*.

(b) Give a brief description of the structure of the *symbol table record* you use.

(c) Improve the *3-address code* of the different *basic blocks* using standard optimization techniques e.g. *constant folding, constant propagation, elimination of common subexpression, strength reduction, elimination of useless instruction* etc. Give a brief explanation in each case. [Do not bother to improvement code across basic block boundaries.]

(d) Assume that `main` is a function and translate the improved *3-address code* sequence to the assembly language program of *x86-64, GNU* (as faithfully possible). Let the offsets (-ve) of different objects in the *activation record (stack-frame)* with respect to the *base pointer* rbp be as follows:

Location	`data`	`n`	`i`	`imin`	`minD`	`j`	`temp`
Offset	400	408	416	424	432	440	448

Registers	*Usage*
eax/rax	*return value from a function*
rdi	*1st argument to a function*
rsi	*2nd argument to a function*
rdx	*3rd argument to a function*
rcx	*4th argument to a function*

3. [10 + 5]

(a) Consider the following CFG. S is the start symbol, $\{S, A, b\}$ is the set of *non-terminals*. Formally justify whether the grammar is SLR, LALR(1), LR(1), or none of these.

$$\begin{aligned} S &\rightarrow aSa \mid Aa \mid bAc \mid Bc \mid bBb \mid bBa \\ A &\rightarrow d \\ B &\rightarrow bBb \mid d \end{aligned}$$

(b) Briefly describe the translation of `switch`-statement used in a programming language.

4. [10 + 5]

(a) Write L-attributed SDD's for the following productions (Boolean expressions and statements have their usual semantics).

$$\begin{aligned} B &\rightarrow B || B \mid B \&\& B \mid !B \mid (B) \mid E \; \texttt{rel} \; E \\ S &\rightarrow \texttt{ASSIGN\_STMT} \mid \texttt{if } B \; S_1 \texttt{ else } S_2 \mid \texttt{do } S_1 \texttt{ while } B \mid \texttt{for(} E_1\texttt{;} \; B\texttt{;} \; E_2\texttt{)} \; S_1 \end{aligned}$$

Assume that the non-terminals have their usual inherited and synthesized attribute e.g. *B.true, B.false, S.next* and *S.code, E.code, B.code.* Make reasonable assumptions about unspecified objects (if necessary).

(b) Draw the syntax tree corresponding to the following code and annotate it using your SDD.

```
do {
if (i < 100 || i > 200) S1 else S2
} while (j > i && j < 500)
```