BRAC UNIVERSITY
Department of Computer Science and Engineering

Examination: Final | Semester: Summer 2024

Duration: 1 Hour 10 Minutes | **Set - A** | Full Marks: 35

CSE 420: Compiler Design

Figures in the right margin indicate marks.

Answer all the questions

COs	Questions	Marks
CO3	1. Consider the following grammar and look at the SLR(1) parse table below:	10

1. $E \rightarrow E + T$
2. $E \rightarrow T$
3. $T \rightarrow T * F$
4. $T \rightarrow F$
5. $F \rightarrow (E)$
6. $F \rightarrow id$

STATE	ACTION						GOTO		
	id	+	*	(	)	$	E	T	F
0	s5			s4			1	2	3
1		s6				acc			
2		r2	s7		r2	r2			
3		r4	r4		r4	r4			
4	s5			s4			8	2	3
5		r6	r6		r6	r6			
6	s5			s4				9	3
7	s5			s4					10
8		s6			s11				
9		r1	s7		r1	r1			
10		r3	r3		r3	r3			
11		r5	r5		r5	r5			

	Show the parsing simulation using **stack** for the input string, (***id*id)+id***	
CO4	2. Consider the following *SDT*. Draw the **Dependency Graph** and show the values of the attribute offset, type, and width next to the appropriate non-terminal nodes of the following input string **int [2][8] a; float b;** $P \rightarrow \quad \{\ offset = 0;\ \}\ D$ $D \rightarrow T\ \textbf{id}\ ;\quad \{\ top.put(\textbf{id}.lexeme,\ T.type,\ offset);\ offset = offset + T.width;\ \}\ D_1$ $D \rightarrow \epsilon$ $T \rightarrow B\ \{\ t = B.type;\ w = B.width;\ \}\ C$ {T.type = C.type; T.width = C.width} $B \rightarrow \textbf{int} \quad \{\ B.type = integer;\ B.width = 4;\ \}$ $B \rightarrow \textbf{float} \quad \{\ B.type = float;\ B.width = 8;\ \}$ $C \rightarrow \epsilon \quad \{\ C.type = t;\ C.width = w;\ \}$ $C \rightarrow [\ \textbf{num}\ ]\ C_1 \quad \{$ C.type = $array(\textbf{num}.value,\ C_1.type);\ C.width = \textbf{num}.value \times C_1.width;\ \}$	10
CO5	3. Construct the **Three Address Code** from the following code: **a = ((c-d)*(-b)+(c+b))-(d*c)+b;** **generateRandom(seed, seedMod5, seedMod9);**	7
CO4	4. Given the following C/C++ code below, suppose the CPU is executing the line '$intermediateResult\ *=\ sqrt(intermediateMultiplier);$' in the code. Draw the content of the active symbol tables at that point. Show the organization of the active symbol tables as a *single list of hash-tables* in your drawing and for each variable in a symbol table, write its name and type. Remember that, in C/C++ each {} group represents a separate scope. That is, there are separate scopes for functions, for, and while loops. You do not need to separate the function header and body into two separate scopes, by the way. (Hint, when you write a statement like '$t\ n\ =\ v$' in C, it	8

means you are defining a new variable with the name n of the type t and assigning it an initial value v)

```
float x = 0.0;
float y = 0.0;
int grid[10][20];

void main(int argCount, char* argValues[]) {
   x = atof(argValues[1]);
   grid[0][0] = atoi(argValues[2]);

   float factor = 1.0;
   int rowLimit = 10;

   for (int r = 0; r < rowLimit; r++) {
      float total = 0.0;
      int colLimit = 20;

      float intermediateMultiplier = 1.0;
      for (int v = 0; v < 5; v++) {
         intermediateMultiplier *= sqrt(x + v);
      }

      for (int c = 0; c < colLimit; c++) {
         int num = grid[r][c];
         float intermediateResult = 1.0;

         for (int k = 0; k < num; k++) {
            intermediateResult *= sqrt(intermediateMultiplier);
            while(k < c) {
                  int  count = 0;
                   count = x + k;
                  }
            int limit = 100;
         }
         total += intermediateResult;
      }

      factor *= total;
   }
}
```