# 18CS604 - COMPILER DESIGN LABORATORY

**Class / Semester:** III CSE C / VI **Date:** 31.03.2021 (FN & AN)

**Time:** 3 Hours **Maximum Marks**: 100

|  |  |
| --- | --- |
| **Register Number** | **Question** |
| 18TUCS201 | Given n names and phone numbers, assemble a phone book that maps friends' names to their respective phone numbers. You will then be given an unknown number of names to query your phone book for. For each name queried, print the associated entry from your phone book on a new line in the form **name=phone Number**; if an entry for name is not found, print Not found instead. |
| 18TUCS202 | Write a program that creates a symbol table mapping letter grades to numerical scores, as in the table below, and then reads from standard input a list of letter grades and computes their average (GPA).   |  |  |  |  |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | | A+ | A | A- | B+ | B | B- | C+ | C | C- | D | F | | 4.33 | 4 | 3.67 | 3.33 | 3 | 2.67 | 2.33 | 2 | 1.67 | 1 | 0 | |
| 18TUCS203 | Consider the following grammar  E → E + T | T  T → T \*F | F  F → a | b  Construct the SLR parsing table and show the moves of the parser for **a + b \* a & a b +** |
| 18TUCS204 | Consider the following grammar  E → E + T | T  T → T \* F | F  F → a | b  Construct LALR parser and show the moves of the parser for a + b \* a & a b + |
| 18TUCS205 | You have a test string S. Your task is to write a pattern using lex rules to match the   1. Octal and hexadecimal numbers. 2. capitalize the given comment |
| 18TUCS206 | Implement the back end of the compiler which takes the three address code and Generate target code for the given program segment:  T1 := p + q  T2 := s \* t  T3 := E - T2  T4 := T1 - T3 |
| 18TUCS207 | Generate the optimized code for the given statement,  while (i<=t+10{  i=i+1;  } |
| 18TUCS208 | Generate the target code for the given postfix expression as "ab+cf\*/e". |
| 18TUCS209 | Construct predictive parsing table for the grammar  S🡪(L) | a  L🡪L,S | S  Also construct the behavior of the parser on the sentence (a,a) for the grammar specified above. |
| 18TUCS210 | Implement the back end of the compiler which takes the three address code and produces the 8086 assembly language instructions that can be assembled and run using an 8086 assembler. Generate target code for the given   program segment:  main()  {  int i,j;  i=4; j=i+5;  } |
| 18TUCS211 | Generate intermediate code for the following code segment  int a,b;  float c;  a=10;  switch(a)  {  case 10: c=1;  case 20: c=2;  } |
| 18TUCS212 | Consider the following grammar  S → AA  A → Aa | b  Construct the CLR parsing table |
| 18TUCS213 | Generate intermediate code for the following code segment along with the Optimized Code  a=2\*b+3\*c |
| 18TUCS214 | Generate Optimized code for the following code segment,  T1 := A + B  T2 := C + D  T3 := A + B  T4 := T1 - T3 |
| 18TUCS215 | Generate Optimized code removing Common sub expression elimination in the following code segment  ((x+y)-((x+y)\*(x-y)))+((x+y)\*(x-y)) |
| 18TUCS216 | Generate YACC Specification to recognize a valid arithmetic expression that uses operator +, - , \* and /. |
| 18TUCS217 | LEX rules to count the no of ‘scanf’ and ‘printf’ statements in a C program. Replace them with ‘readf’ and ‘writef’ statements respectively. |
| 18TUCS218 | Design a Shift reduce parser to validate any input string for the given language specification   (Grammar). Parse the given input string “d\*d/d”.  E🡪E/E  E🡪E\*E  E🡪d |
| 18TUCS219 | Consider the grammar, Design a Shift reduce parser to validate any input string for the given language specification,  E → E+T | T  T → T\*F | F  F → (E) | id  Show the sequence of moves made by the shift-reduce parser on the input “id + id \*id” using stack. |
| 18TUCS220 | Generate the assembly language for the statement W:= (A + B) + (A + C) + (A + C). |
| 18TUCS221 | Generate Optimized code for the following code segment,  T1 := A + B  T2 := C + D  T3 := A + B  T4 := T1 - T3 |
| 18TUCS222 | Generate intermediate code for the following code segment  int a,b;  float c;  a=10;  switch(a)  {  case 10: c=1;  case 20: c=2;  } |
| 18TUCS223 | Generate Optimized code removing Common sub expression elimination in the following code segment  ((x+y)-((x+y)\*(x-y)))+((x+y)\*(x-y)) |
| 18TUCS224 | Generate intermediate code for the following code segment along with the Code Optimization Techniques  if(a>b)  x=a+b  else  x=a-b  where a & x are of real and b of int type data. |
| 18TUCS225 | Design a Predictive Parser for a language specification to eliminate left recursion and left factoring.  E → E + T | T  T → T \* F | F  F → a | b  Show the moves of the parser for the string “a + b \* a” |
| 18TUCS226 | Design a Predictive Parser for a language specification to eliminate left recursion and left factoring.  *stmt* → if *expr* then *stmt*  | if *expr* then *stmt* else *stmt*  *expr→ b* |
| 18TUCS227 | Implement the back end of the compiler which takes the three address code and Generate target code for the given program segment:  T1 := p + q  T2 := s \* t  T3 := E - T2  T4 := T1 - T3 |
| 18TUCS228 | Generate intermediate code for the following code segment along with the Optimized Code Y=A+B\*x+C\*(x\*\*2)+D\*(x\*\*2) |
| 18TUCS229 | Consider the following grammar  S → AA  A → Aa | b  Construct the CLR parsing table |
| 18TUCS230 | Consider the grammar  E🡪E-E  E🡪E\*E  E🡪id  Perform Shift Reduce Parsing for the string id-id\*id |
| 18TUCS231 | Generate intermediate code for the following code segment along with the Optimized Code Y= C\*(x\*\*2)+D\*(x\*\*2) |
| 18TUCS232 | Generate intermediate code for the following code segment  int a,b;  float c;  a=10;  switch(a)  {  case 10: c=1;  case 20: c=2;  } |

**INTERNAL EXAMINER EXTERNAL EXAMINER**