| | |
|---|---|
| BRANCH | CSE |
| SEMESTER | VI |

**SAGAR INSTITUTE OF SCIENCE & TECHNOLOGY**
**DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING**

**QUESTION BANK**

**NAME OF THE FACULTY: Bhavana Gupta**

**SUBJECT/CODE : COMPILER DESIGN/CS-603(C)**

| S. No. | QUESTIONS (UNIT-1) | CO's |
|---|---|---|
| 1 | Show the various phases of compiler? How phases of compilation converts the statement<br>Position=initial+rate*60 | CO-1<br>3(Apply) |
| 2 | Interpret the concept of bootstrapping. | CO-1<br>2(Understand) |
| 3 | Explain the following in brief:<br>(i)Input buffering<br>(ii) Function of lexical analyzer | CO-1<br>1(Remember) |
| 4 | Define the LEX? Describe auxiliary definitions and translation rules for LEX with suitable example. | CO-1<br>1(Remember) |
| 5 | Identify the number of tokens in the given C statements:<br>(i) printf("i=%d, and i=%x", i,&i);<br><br>(ii) main()<br>{<br>int<br>x=1<br>0;<br>x=x<br>+y<br>+z;<br>} | 3(Apply) |
| 6 | Listthe issues in lexical analysis? Classify the tokenrecognition with suitable example. | CO-1<br>1(Remember) |
| 7 | Construct minimal DFA that accept all binary numbers which are divisible by 4 | CO-1<br>3(Apply) |
| 8 | Explain how the recognition of tokens is done using transition graph. | CO-1<br>2(Understand) |
| 9 | Construct a minimal DFA that accept a language containing all binary string and that starts and end with same symbol | CO-1<br>3(Apply) |
| 10 | What are the various classifications of compiler? | CO-1<br>2(Understand) |

| | | |
|---|---|---|
| 11. | Explain the process of lexicalanalyzer with the help of diagram. | CO-1<br>2(Understand) |
| 12. | Explain left recursion and show how it is eliminated. Describe the algorithm used for eliminating left recursion | CO-1<br>2(Understand) |
| 13 | Illustrate the concept of cross compiler. Give an example | CO-1<br>2(Understand) |
| 14 | Explain Finite State Machine with its limitation and applications. | CO-1<br>2(Understand) |
| 15 | Illustrate the concept of compiler and interpreter. | CO-1<br>2(Understand) |

| **S. No.** | **QUESTIONS (UNIT-2)** | **CO's** |
|---|---|---|
| 1 | Show whether the following grammar is ambiguous or not. Also, construct the parse tree using LMD and RMD.<br>E->E+E/E*E/(E)/id<br>input string: **id1+id2*id3** | CO-2<br>3(Apply) |
| 2 | Examine whether the given grammar is LL(1) or not?<br>S->iEtSS'/a<br>S'->eS/E<br>E->b | CO-2<br>3(Apply) |
| 3 | . Examine that the following grammar is SLR(1) or not.<br>S-AaBb/BbBa<br>A->ε<br>B->ε | CO-2<br>3(Apply) |
| 4 | Write the algorithm for FIRST and FOLLOW. Consider the grammar:<br>**S->ACB/CbB/Ba**<br>**A->da/BC**<br>**B->g/ε**<br>**C->h/ε**<br>construct FIRST and FOLLOW | CO-2<br>3(Apply) |
| 5 | Construct LR(0) parsing table for the following grammar?<br>**E-> E+T/T**<br>**T-> T*F/F**<br>**F-> (E)/id** | CO-2<br>3(Apply) |
| 6 | Contrast on operator precedence parsing method using sutable example. | CO-2<br>3(Apply) |
| 7 | Examine the given grammar is SLR(1) or not?<br>**E-> E+T/T**<br>**T-> T*F/F**<br>**F-> (E)/id** | CO-2<br>3(Analyze) |
| 8 | . Illustrate the working process of YACC. | CO-2<br>2(Understand) |
| 9 | Construct the CLR(1) parser for the given grammar.<br>E->BB<br>B->.cB/d | CO-2<br>3(Apply) |
| 10 | Consider the following grammar G: | |

| | | |
|---|---|---|
| | S->AB/d<br>A->aA/b B->bB/c<br>The grammaris:<br>a) LL(1) grammar and notLR(0)<br>b) LL(1) andLR(0)<br>c) Not LL(1) butLR(0)<br>d) Neither LL(1) nor LR(0)<br>Also justify the correct option. | CO-2<br>4(Analyze) |
| 11. | Demonstrate the syntax directed translation on the basis of syntax directed definition and translation scheme? | CO-2<br>3(Apply) |
| 12. | Explain translation rule for expression **2+5*4** according to the syntax. | CO-2<br>2(Understand) |
| 13. | Interpret the causes of backtracking in top down parser? Explain with an example. | CO-2<br>3(Apply)<br>2(Understand) |
| 14 | Illustrate the LL (1) grammar. Classify the properties of LL (1) grammar. | CO-2<br>2(Understand) |
| 15 | What is CFG? Explain elimination of CFG with examples. | CO-2<br>1(Remember)<br>2(Understand) |
| **S. No.** | **QUESTIONS (UNIT-3)** | **CO's** |
| 1 | Illustrate the concept of type checking and type conversion with the help of an example. | CO-3<br>3(Apply) |
| 2 | Demonstrate the various strategies of symbol table creation and organization. | CO-3<br>2(Understand) |
| 3 | Classify the different storage allocation strategies with the help of suitable example. | CO-3<br>2(Understand) |
| 4 | What is run time environment? What are the important elements of runtime environment? How it is controlled in a program that is compiled? | CO-3<br>1(Remember)<br>2(Understand) |
| 5 | Interpret the symbol table in detail. | CO-3<br>2(Understand) |
| 6 | Define an activation record? With the help of diagram show important fields in an activation record. | CO-3<br>1(Remember) |
| 7 | Examine the error detection and recovery phase. | CO-3<br>4(Analyze) |
| 8 | Show the static and dynamic storage allocations. | CO-3<br>1(Remember) |
| 9 | Classify the difficulties faced by memory allocation for variable length requirements? Under what circumstances does external fragmentation happen? | CO-3<br>2(Understand) |
| 10 | Write short notes on:<br>a. Parameter Passing<br>b. Polymorphic function | CO-3<br>1(Remember) |
| 11 | Contrast the step to construct the predictive parser table. | CO-3<br>2(Understand) |
| 12 | Explain Register allocation and assignment with suitable example. | CO-3 |

| | | |
|---|---|---|
| | | 2(Understand) |
| 13 | Explain the concept of static, stack and heap allocation. | CO-3<br>2(Understand) |
| 14 | What are the procedure calling and returning sequences? Explain the sequence of action in each of them. | CO-3<br>1(Remember)<br>2(Understand) |
| 15 | Classify the implicit type conversion and explicit type conversion. | CO-3<br>2(Understand) |

| S. No. | QUESTIONS (UNIT-4) | CO's |
|---|---|---|
| 1 | Construct DAG for the basic block whose code is given below:<br>**D:=B*C**<br>**E:=A+B**<br>**B:=B*C**<br>**A:=E-D** | CO-4<br>3(Apply) |
| 2 | Demonstrate the quadruple, triple, and indirect triple with the help of example. | CO-4<br>2(Understand) |
| 3 | Translate the following into three address code:<br>**begin**<br>**PROD :=0;**<br>**I:=1;**<br>**do**<br>**begin**<br>**PROD:=PROD+A[i]*B[i];**<br>**I:=I+1;**<br>**end**<br>**while I<=10**<br>**end** | CO-4<br>3(Apply) |
| 4 | Analysis the possible causes of dead code? Explain with an example how the compiler can catch the presence of a dead code. | CO-4<br>4(Analyze) |
| 5 | Write down the three address code for the following switch statement:<br>**swith(ch)**<br>**{**<br>**case 1: c:=a+b;**<br>**break;**<br>**case 2: c:=a-b;**<br>**break;**<br>**}** | CO-4<br>3(Apply) |
| 6 | Generate three address code for the following expression:<br>**a / (b+c) ^ d + e*f**<br>also represent the above expression in quadruple, triple, indirect triple. | CO-4<br>3(Apply) |
| 7 | Explain briefly:<br>a. Three Address Code<br>b. DAG | CO-4<br>2(Understand) |
| 8 | Write notes on peephole optimization. | CO-4 |

| | | |
|---|---|---|
| | | 1(Remember) |
| 9 | Explain registers allocation strategies with the help of an example. | CO-4<br>2(Understand) |
| 10 | Explain Back Patching in detail. | CO-4<br>2(Understand) |
| 11 | Construct the syntax tree and DAG for the following expression<br>A:=B*-C+B*-C | CO-4<br>3(Apply) |
| 12 | Interpret the different issues in the design of code generator. | CO-4<br>3(Apply) |
| 13 | With an example explain the various format of intermediate code. | CO-4<br>2(Understand) |
| 14 | Illustrate the Back patching techniques. | CO-4<br>2(Understand) |
| 15 | What are the DAGs and how are they useful in implementing transformations on basic blocks? | CO-4<br>1(Remember) |

| **S. No.** | **QUESTIONS (UNIT-5)** | **CO's** |
|---|---|---|
| 1 | What is global data flow analysis? What is its use in code optimization? | CO-5<br>1(Remember) |
| 2 | What is loop optimization Explain? | CO-5<br>1(Remember) |
| 3 | Illustrate the common sub expression elimination copy propagation and transformation for moving loop invariant computations in detail. | CO-5<br>2(Understand) |
| 4 | Examine the algorithm for global common sub expression elimination. | CO-5<br>4(Analyze) |
| 5 | Classify any two data flow properties used by target code generator for producing efficient code. | CO-5<br>2(Understand) |
| 6 | Show various issues in design of code generator. | CO-5<br>2(Understand) |
| 7 | Explain the concept of dead code elimination. | CO-5<br>2(Understand) |
| 8 | Analyze the possible causes of a dead code? Explain with an example how the Compiler can catch the presence of a dead code. | CO-5<br>4(Analyze) |

| | | |
|---|---|---|
| 9 | Explain the following:<br>i)Strength Reduction<br>ii)Variable propagation | CO-5<br>2(Understand) |
| 10 | Define the following:<br>i)Natural loops<br>ii)Inner loop | CO-5<br>1(Remember) |
| 11 | Mention the criteria for code improving transformations. | CO-5<br>1(Remember) |
| 12 | Survey how copy propagation can be done using data flow equation. | CO-5<br>3(Apply) |
| 13 | Classify the properties of optimizing compiler. | CO-5<br>2(Understand) |
| 14 | Demonstrate the various code optimization techniques in detail. | CO-5<br>2(Understand) |
| 15 | Explain any two data flow properties used by target code generator for producing efficient code. | CO-5<br>2(Understand) |