|  |  |
| --- | --- |
| **SESSION** | **FEB 2024** |
| **PROGRAM** | **BCA** |
| **SEMESTER** | **V** |
| **COURSE CODE & NAME** | **DCA3141 - COMPILER DESIGN**  |
|  |  |
|  |  |

**SET-I**

**1. Explain different phases of Compiler with a suitable example.**

**Ans1.**

Compiler design is a fundamental area of computer science that focuses on converting high-level programming languages into machine code or lower-level languages. A compiler goes through several phases to accomplish this task, each with its unique role in the process of understanding, analyzing, and finally transforming the source code into target code. Here's a detailed explanation of the different phases of Its Half solved only

Buy Complete from our online store

<https://smuassignment.in/online-store/>

MUJ Fully solved assignment available for**session FEB 2024.**

Lowest price guarantee with quality.

Charges**INR 198 only per assignment.**For more information you can get via mail or Whats app also

Mail id is aapkieducation@gmail.com

Our website www.smuassignment.in

After mail, we will reply you instant or maximum

1 hour.

Otherwise you can also contact on our

whatsapp no 8791490301.

**2. Calculate the First(S) of given below Grammar:**

**S->ABC**

**A->a/b/ɛ**

**B->c/d/ ɛ**

**C->e/f/ ɛ**

**Also calculate Follow(S) of given below Grammar:**

**S->ACD**

**C->a/b**

Ans 2.

Calculating the **First** and **Follow** sets of a given grammar are key steps in constructing parsers, especially for LL(1) parsers. These sets help the parser decide which production rule to apply based on the current input symbol.

**Calculating First(S)**

The **First** set of a

**3. Create the Parsing table of following Grammar:**

**A->CB**

**B->xCB/ɛ**

**C->DE**

**E->yDE/ ɛ**

**D->z/(A)**

**Ans 3.**

Creating a parsing table is a crucial step in constructing an LL(1) parser. This table guides the parser in deciding which production to use based on the current input symbol and the top of the parsing stack (which holds grammar symbols). Let's construct the parsing table for the given

**Set-II**

**4. Discuss conceptual view of Syntax Direct Translation. Elaborate the method for bottom-up evaluation of inherited attributes.**

**Ans 4.**

Syntax-Directed Translation (SDT) is a method used in compilers to perform translations of programming languages based on their syntax. It essentially ties the process of parsing with translation, utilizing the structure of the source program defined by its grammar to direct the translation process. SDT is often implemented using a parse tree, where each node represents a grammar

Top of Form

**5. Discuss and compare different storage allocation strategies. List different types of scoping rules for non-local name**

**Ans 5.**

Storage allocation strategies in programming languages and compilers play a crucial role in managing memory during the execution of programs. These strategies determine how and when memory for variables is allocated and deallocated. The choice of strategy impacts the efficiency, complexity, and runtime behavior of programs. Additionally, scoping rules define how programs identify the

Top of Form

**6. Classify different Storage allocation strategies. Differentiate between Tail-call optimization and tail-call recursion.**

**Ans 6.**

Storage allocation strategies in programming and compiler design are pivotal for managing memory during program execution. These strategies dictate how memory for variables and data structures is allocated, used, and deallocated, significantly affecting program performance and efficiency. Below, we'll classify these strategies and then distinguish between tail-call