**Assignment 1**

|  |  |
| --- | --- |
| **Course** | **Course Instructor** |
| ***Compiler Construction***   |  | | --- | |  | | Faryal Saud |

**Assignment Guidelines:**

1. Deadlines should be kept in mind. No extension will be given in any case. No late submission will be accepted.
2. This is an individual assignment. **PLAGIARISM IS NOT ACCEPTABLE AT ALL!** Zero marks will be given in case of plagiarism.
3. Deadline: February 07, 2025 in class (Hard Form) and scanned copy on Google Classroom.

|  |
| --- |
| **Question no 1:** |
| **Explain the following with example:**   1. Explain the difference between a compiler and an interpreter. 2. What is the difference between pass and a phase? Explain the design of a compiler   with phases.   1. Explain the different translators required for converting a high-level language to an   executable code. |

|  |
| --- |
| **Question no 2:** |
| Give the output of each phase of the compiler for the following source text:  for (j = 0; j < 5; j++)  sum = sum + j; |

|  |
| --- |
|  |

|  |
| --- |
| **Question no 3:** |
| Perform the lexical analysis for the following C program and find tokens.  main()  {  int a[3], t1t2;  t1=2;  a[0]=1; a[1]=2; a[t1]=3;  t2 = -(a[2]+t1\*6)/(a[2]-t1);  if t2>5  print(t2);  else  {  int t3; t3=99; t2=-25;  print(-t1 +t2\*t3); /\*this is a comment on 2 lines \*/  } endif  } |

|  |
| --- |
| **Question no 4:** |
| Perform the lexical analysis for the following C program and find tokens.  main()  {  int i, j;  while(i<100)  {  i=i+10;  printf(“%d\n”,&i);  }  } |

|  |
| --- |
| **Question no 5:** |
| Divide the following code:  float limitedSquare(x) float x;  /\* returns x-squared, but never more than 100 \*/  return (x<=-10.0||x>=10.0)?100:x\*x  into appropriate lexemes. Which lexemes should get associated lexical values? What should those values be? |

|  |
| --- |
| **Question no 6:** |
| Given the following code snippets:  **(i)**  int a;  float b;  a = 5;  b = 3.14;  **(ii)**  void foo (int x, float y) {  int z;  z = x + y;  }  Draw the symbol table that tracks the variables. Include the types, scopes, and any other relevant information. |

|  |
| --- |
| **Question no 7:** |
| Construct an Abstract Syntax Tree (AST) for the following arithmetic expression and explain its structure:  **result = (a + b) \* (c - d) / 2.** |

***CODING PART***

|  |
| --- |
| **Question no 8:** |
| ****Tokenization and Classification****  * Write a **C program** to recognize and count the number of **keywords, identifiers, constants, and special symbols** in a given C program file. * Implement a **C program** that takes an arithmetic expression as input and extracts all **operators, operands, and parentheses** separately. |

|  |
| --- |
| **Question no 9:** |
| **Comment and Whitespace Handling**   * Write a **C program** to remove all **single-line (//) and multi-line (/\* ... \*/) comments** from a given C source file. * Implement a **C program** to count the **number of spaces, tabs, and newlines** in an input file. |

|  |
| --- |
| **Question no 10:** |
| Write a **C Program** to Read Numbers **(Integer or Floating-Point)** from an Input File and add them. |

***Bonus Task:***

Design a lexical analyzer generator that can handle the following sample input. The lexical

analyzer should ignore redundant spaces, tabs, newlines, and comments.

**Input:**

**main()**

**{**

**int x, y;**

**if (x > y)**

**printf(“x is greater”);**

**else**

**y=10; /\* this is just a comment \*/**

**}**