|  |  |
| --- | --- |
| **Assignment # 2**  Theory of Programming Languages | |
| Student Name | **Qasim Ali** |
| Registration No | MSCS – **3483** |
| Date | 21 Nov 2019 |
| Assignment | Solve Mid Term Paper |

|  |
| --- |
| **Question 1a. What is non-procedural language?** |
| In the non-procedural languages, the user has to specify only “what to do” and not “how to do”. It is also known as an applicative or functional language. It involves the development of the functions from other functions to construct more complex functions.  **Examples.** SQL, PROLOG, LISP.  Difference between procedural and nonprocedural programming language:-   |  |  | | --- | --- | | It is command-driven language. | It is a function-driven language | | It works through the state of machine. | It works through the mathematical functions. | | Its semantics are quite tough. | Its semantics are very simple. | | It returns only restricted data types and allowed values. | It can return any datatype or value | | Overall efficiency is very high. | Overall efficiency is low as compared to Procedural Language. | | Size of the program written in Procedural language is large. | Size of the Non-Procedural language programs are small. | | It is not suitable for time critical applications. | It is suitable for time critical applications. | | Iterative loops and Recursive calls both are used in the Procedural languages. | Recursive calls are used in Non-Procedural languages. | |
| **Question 1b. Which features of a language have a significant impact on reliability of a program written in a given language? Briefly explain each one of them.** |
| **Type checking**  Testing for type errors  **Exception handling**  Intercept run-time errors and take corrective measures  **Aliasing**  Presence of two or more distinct referencing methods for the same memory location  **Readability and writability**  A language that does not support “natural” ways of expressing an algorithm will require the use of “unnatural” approaches, and hence reduced reliability |
| **Question 1c. Why it is important to have some background knowledge in programming language design even though you may not be designing a programming language?** |
| * Increased ability to express ideas * Improved background for choosing appropriate languages * Increased ability to learn new languages * Better understanding of significance of implementation * Better use of languages that are already known * Overall advancement of computing |
| **Question 2a. Identify the lexemes and tokens of the following statement**  **Index = 2 \* count + 17;** |
| |  |  | | --- | --- | | Lexeme | Token | | Index | Identifier | | = | Equal\_sign | | 2 | Int\_literal | | \* | Mult\_op | | Count | Identifier | | + | Plus\_op | | 17 | Int\_literal | | ; | Semicolon | |

|  |
| --- |
| **Question 2b. Prove the above grammer is a BNF grammer.** |
|  |
| **Question 3a. What are the major design issues related to string data type? Explain with suitable examples.** |
| The two most important design issues that are specific to character string types are the following:  • Should strings be a special kind of character array or a primitive type?  • Should strings have static or dynamic length? |
| **Question 3b. Why bound checking in array is important? Explain** |
| Array bound checking refers to determining whether all array references in a program are within their declared ranges. This checking is critical for software verification and validation because subscripting arrays beyond their declared sizes may produce unexpected results, security holes, or failures. |
| **Question 3c. What are the benefits of enumerated data types?** |
| Enumeration types can provide advantages in both readability and reliabil- ity.  Readability is enhanced very directly.  Named values are easily recognized, whereas coded values are not. |
| **Question 4a. Assume you want to design a new programming language and wanted to decide on the variables and variable declaration in the new language. For your language,**   1. **Will you prefer case sensitive variable names or not? Why?** 2. **Will you prefer explicit or implicit declaration? Why?** |
| 1. I will prefer case sensitive variable names due to better readability. It is being used in all advanced languages to include php, JavaScript etc. 2. I will prefer implicit declaration to assist programmer write programs faster. It also helps in handling more data types in same variable. Construction of such a programming language is more time consuming. This is system heavy approach. |
| **Question 4b. Consider the program shown in paper, answer following**   1. **The above program has one error. Identify it? What is the reason for this error?** 2. **If the above program is run after correcting the error, what will be the values of w, p and w will be printed?** 3. **How multiple declaration of a variable in different program blocks affects reasiblity and writability fo the language? Explain your answer.** |
| 1. error: ‘p’ was not declared in this scope cout<<p; 2. w inside for 8;   p 16  8   1. Multiple declaration of a variable is not recommended as it greatly reduces the writability of a program. It also confuses the developer while de bugging the code. Readability of a program is not affected by multiple declaration of a variable in a program. |