A Quarterly E-Magazine
ISSN 2582-5895
Volume 9 Issue 1 Month 1 Year 2026

Communications of Adhikari Academy of Learning in Computer Science and Information Technology

Adhikari Academy of Learning Private Limited

Communications of Adhikari Academy of Learning
in
Computer Science and Information Technology

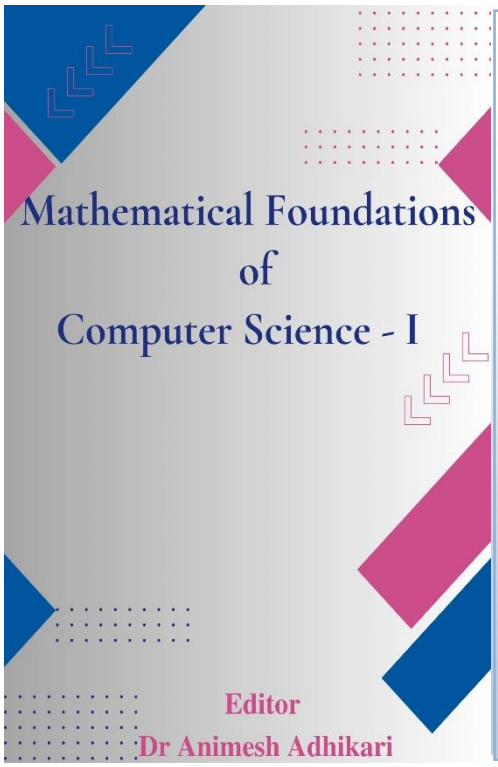
EDITORS

Published by Adhikari Academy of Learning Private Limited, F-2, Kurtarkar Nagari-30, Shantinagar, Ponda, Goa - 403401, India

News

We published the book *Mathematical Foundations of Computer Science – I* (Second Edition) recently. Details of the book are given as follows.

The book includes interesting problems / solutions on set, function, relation, propositional calculus and predicate calculus. The selection of questions / problems is aimed at providing different skills to learners. This book is also effective for students other than Computer Science, who have taken discrete mathematics as a paper of their studies. The second edition of the book contains 417 questions / solutions.



Editor(s): Dr Animesh Adhikari, Ph D

Biography: Dr Animesh Adhikari is Founder and Editor of Adhikari Academy of Learning Private Limited, Goa, India. Animesh received Master of Technology and Ph D degrees, both in Computer Science, from Indian Statistical Institute and Goa University, respectively.

Animesh has been working in both academia and software industry since 1993. In the past, he served to K L University, Guntur; Chowgule College, Margao; and Price Waterhouse Associates, Kolkata.

Animesh published three research monographs on *Data and Knowledge Engineering* with Springer, Germany, and twenty research papers in international journals and international conferences. He published twenty-nine books on various aspects of Computer Science and Information Technology with Adhikari Academy of Learning, Goa. Animesh also published a few derived books such as Mathematics for Computer Science and Information Technology, and GATE - Computer Science and Information Technology.

For more details about the book, please visit book webpage.

Table of Contents

Keywords

Keyword	Question number	Page number
dynamic testing	2	2
static testing	2	2
system catalog	3	3

1. Using a stack, write a C program to check whether a string of parentheses is balanced or not.

Communicated by Jhimli Adhikari

Answer: A program written in C language for the above task is given below.

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

#define MAX_SIZE 50 // maximum size of stack

char stack[MAX_SIZE]; // global stack variable
int top = -1; // with stack top initialization

void push(char c) { // to add a character onto the stack
    if (top == MAX_SIZE - 1) {
        printf("Stack is overflowed!\n");
        return;
    }
    top++;
    stack[top] = c;
}

char pop() { // to delete a character from the stack
    if (top == -1) {
        printf("Stack is empty!\n");
        return ' ';
    }
    char c = stack[top];
    top--;
    return c;
}

int is_matching_pair(char char1, char char2) { // to check if two characters
form a matching pair of parentheses
    if (char1 == '(' && char2 == ')') {
        return 1;
    } else if (char1 == '[' && char2 == ']') {
        return 1;
    } else if (char1 == '{' && char2 == '}') {
        return 1;
    } else {
        return 0;
    }
}

int is_balanced(char* text) { // to check if the expression is balanced
    int i;
    for (i = 0; i < strlen(text); i++) {
        if (text[i] == '(' || text[i] == '[' || text[i] == '{') {
            push(text[i]);
        } else if (text[i] == ')' || text[i] == ']' || text[i] == '}') {
            if (top == -1) {
```

```
                return 0; // if no opening bracket is present
            } else if (!is_matching_pair(pop(), text[i])) {
                return 0; // if closing bracket does not match the last
opening bracket
            }
        }
    }
    if (top == -1) { // the expression is balanced
        return 1;
    } else { // the expression is not balanced
        return 0;
    }
}

int main() { // function for the given task
   char expr[MAX_SIZE];
   printf("Input an expression containing parentheses: ");
   scanf("%s", expr);
   if (is_balanced(expr)) {
       printf("\nThe given expression is balanced.");
   } else {
       printf("\nThe given expression is not balanced.");
   }
   return 0;
}
```

A few executions with input / output are given below.

```
Input an expression containing parentheses: ((()))[{}]
The given expression is balanced.
---
Input an expression containing parentheses: {(})[]
The given expression is not balanced.
---
Input an expression containing parentheses: [[[]]{}]()
The given expression is balanced.
---
```

NOTE: We have used the following online C compiler to execute the above program:
https://www.programiz.com/c-programming/online-compiler/

2. What is the difference between static and dynamic testing in software development?

Communicated by Jhimli Adhikari

Answer: Here we discuss a few points that distinguish between static and dynamic testing.

In *static testing*, we analyze software components like code, design documents, and requirements without running the program itself. It is like examining blueprints for a house to identify potential issues before construction begins. Static testing helps prevent defects by catching errors early on. Examples include code reviews, where developers inspect each other's code for mistakes, and walkthroughs, where a team discusses the design and logic of the software.

Dynamic testing method involves executing the actual software code to see how it behaves under various conditions. It is like testing the built house to ensure everything functions as planned. Dynamic testing helps find and fix defects in the running application. Examples include unit testing (testing individual modules of code), integration testing (testing how different modules work together), and system testing

(testing the entire software system with real-world data).

3. What information is stored in system catalogs? State its functions.

Communicated by Animesh Adhikari

Answer: System catalogs store metadata, which is data about other data, including schema information for tables, views, and columns, as well as details on indexes, constraints, users, privileges, and internal performance statistics like query history and object growth. We preset here more details about this information.

Schema and object definitions:

- *Table and column definitions*: Information about tables, their columns, data types, and whether they are nullable.
- *Index structures*: Details on the structure and storage of indexes.
- *Views, triggers, and stored procedures*: Metadata for other database objects like views, stored procedures, and triggers.
- *Constraints*: Information on table constraints and referential integrity rules.

Security and access control:

- *User accounts*: Details about user accounts and their default settings.
- *Privileges*: Information about user privileges and security settings.

Performance and operational data:

- *Performance statistics*: Data on system performance, including query statistics, performance data, and object sizing.
- *Internal settings*: Internal database settings and configurations.
- *User sessions*: Information on active user sessions.
- *Auditing information*: Records of database activities for auditing purposes.

Other information:

- *Statistical data*: Data like the number of tuples (rows), blocks, and the size of a relation (table).
- *Internal database state*: Details on the internal state and operations of the database.
- *Data dictionary views*: User-friendly views that provide access to the metadata stored in the system catalogs.

Functions of systems catalogs include the following activities:

- *Self-description*: The system catalogs act as a data dictionary, containing all the information the database needs to know about itself.
- *Database management*: It's crucial for the RDBMS to manage the structure, content, and operation of the database.
- *User access*: Users can query system catalogs to gain insights into the database's structure, performance, and internal workings.

4. A graph has 26 vertices and 58 edges. There are five vertices of degree 4, six vertices of degree 5, and seven vertices of degree 6. If the remaining vertices all have the same degree, what is this degree?

Communicated by Animesh Adhikari

Answer: There are $26 - 5 - 6 - 7 = 8$ vertices remaining. They all have the same degree. Let the degree of each such vertex be x.

Now, each edge contributes 2 degrees. Then, total degrees $= 2 \times 58 = 116$. (1)

Using degree of a vertex, we get total degrees $= 5 \times 4 + 6 \times 5 + 7 \times 6 + 8 \times x$

$= 92 + 8x$ (2)

Using (1) and (2), we get $116 = 92 + 8x$, or, $x = 3$.

Thus, the degree of the remaining eight vertices is 3.

5. Consider the following functions.

$f(x) = 3n^{\sqrt{n}}$

$g(n) = 2^{\sqrt{n}\log_2 n}$

$h(n) = n!$

Which of the following is true?

(a) $h(n)$ is $O(f(n))$ (b) $h(n)$ is $O(g(n))$

(c) $g(n)$ is not $O(f(n))$ (d) $f(n)$ is $O(g(n))$

Communicated by Animesh Adhikari

Answer: $g(n) = 2^{\sqrt{n}\log_2 n}$

or, $\log_2(g(n)) = \log_2(2^{\sqrt{n}\log_2 n})$, since $\log_2(x)$ is a monotonic increasing function of x

or, $\log_2(g(n)) = \sqrt{n}\log_2 n \times \log_2(2) = \sqrt{n}\log_2 n$

or, $\log_2(g(n)) = \log_2 n^{\sqrt{n}}$

or, $g(n) = n^{\sqrt{n}}$

$\therefore\ f(x) = 3n^{\sqrt{n}} = O(n^{\sqrt{n}}) = O(g(n))$

So, option (d) is correct.

Contact Emails

01 **Animesh Adhikari**

animeshadhikari@yahoo.com

02 **Jhimli Adhikari**

jhimli_adhikari@yahoo.co.in

