|  |  |  |  |
| --- | --- | --- | --- |
| PICTLOGO | PUNE INSTITUTE OF COMPUTER TECHNOLOGY  PUNE - 411043 | | |
| Department of Electronics & Telecommunication | | |
| ASSESMENT YEAR: 2024-2025 | CLASS: SE | |
| SUBJECT: DATA STRUCTURES | | |
| **EXPT No:** | LAB Ref: SE/2024-25/ | | Starting date: |
|  | Roll No:22168 | | Submission date: |
| **Title:** | **Exp-06: Arithmetic Operations on Polynomial** | | |
| **Problem Statement** | Write a program in C to illustrate the addition of two polynomials using a  linked list. Test various input for addition, subtraction, and multiplication  of two polynomial by following all mathematic rules | | |
| Refer lab manual for below | | | |
| **Prerequisites:** | Software - DEVC++ IDE/GCC | | |
|  | | |
|  | | |
|  | | |
| **Objectives:** | 1. To learn how to represent polynomial using array and linked list. | | |
| 1. Implement polynomial using linked data structure. | | |
| 1. Perform and verify mathematical operation on polynomial. | | |
|  | | |
| **Theory:** | | | |
|  | Write algorithm and besides it example for each, attach both side ruled pages in case required  On Ruled Pages | | |
| **Flow-chart**  **ERRORS** | **(Extra sheet may be attached)**  **Here are some errors :**  ** Memory Allocation Errors: malloc may fail, leading to NULL pointers. Check allocation success before use.**  ** Incorrect Input Handling: Non-numeric input can cause crashes. Validate input types when reading data.**  ** Division by Zero: Zero coefficients can cause errors. Skip zero terms during calculations.**  ** Incorrect Polynomial Order: Nodes may be inserted in the wrong order, affecting results. Ensure nodes are sorted by exponents.**  ** Missing Exponent Handling: Terms with the same exponents might not combine correctly, causing wrong results. Ensure proper merging of terms.**  ** Memory Leaks: Forgetting to free allocated memory can lead to memory leaks. Implement functions to free the linked lists.**  ** Segmentation Faults: Occur due to null pointer dereferencing. Check pointers before access.**  ** Infinite Loops: Caused by incorrect pointer updates during list traversal. Ensure correct loop termination.**  ** Incorrect Subtraction: Mishandling of negative coefficients can lead to errors. Verify the logic for subtracting terms.**  ** Overflow/Underflow of Coefficients: Coefficients may overflow if too large. Use appropriate data types or handle overflow explicitly.**  ** Unintended Polynomial Terms: Incorrect handling can create extra terms. Properly combine terms and remove zero coefficients.**  ** Handling Zero Polynomials: Empty polynomials may cause crashes. Add checks to handle and display zero terms appropriately.** | | |
| **REMEDY** | 1. **Check if memory allocation is successful (malloc not NULL).** 2. **Validate input types before processing; ensure only numbers are accepted.** 3. **Skip zero terms during calculations.** 4. **Ensure nodes are inserted in descending order of exponents.** 5. **Combine terms with the same exponents correctly.** 6. **Implement functions to free all nodes after use to avoid memory leaks.** 7. **Check pointers for NULL before dereferencing.** 8. **Verify proper loop termination and pointer updates.** 9. **Handle negative coefficients accurately during subtraction.** 10. **Use larger data types or add checks to handle coefficient overflow.** 11. **Properly combine terms and remove zero-coefficient terms.** 12. **Add checks for zero or empty polynomials and handle them appropriately by displaying "0".**   **Top of Form**  **Bottom of Form** | | |
|  | | | |
| **CONCLUSION:** | | | |
|  | **The program demonstrates polynomial addition, subtraction, and multiplication** | | |
| **using linked lists, allowing efficient manipulation of polynomial terms. Proper error** | | |
| **handling, such as checking memory allocation, validating input, and managing** | | |
| **terms correctly, ensures the program runs reliably. Addressing these potential** | | |
| **errors enhances the program's robustness and improves skills in handling complex** | | |
| **data structures.** | | |
|  | | |
| **REFERENCES: refer lab manual for the same** | | | |
|  |  | | |
| **1. Seymour Lipschutz, Data Structure with C, Schaum’s Outlines, Tata** | | |
| **McGrawHill** | | |
| **2. Yedidyah Langsam – Data structures using C and C++ - PHI Publications.** | | |
| **3. E Balgurusamy - Programming in ANSI C, Tata McGraw-Hill** | | |
|  | | |

|  |  |  |  |
| --- | --- | --- | --- |
| **Continuous Assessment for DS AY: 2024-25** | | | |
| **RPP (5)** | **SPO (5)** | **Total (10)** | **Signature:** |
|  |  |  | **Assessed By: Dr. V. B. Vaijapurkar** |
| **Start date** | **Submission date** | | **Date:** |
|  |  | |  |
| **\*Regularity, Punctuality, performance**  **\*Submission, Presentation, orals** | | | |