Course Code	**:**	**BCSL-032**
Course Title	**:**	**C++Programming Lab**
Assignment Number	**:**	**BCA(III)/L-032/Assignment/2024-25**
Maximum Marks	**:**	**50**
Weightage	**:**	**25%**
Last date of Submission	**:**	**31st October,2024(for July session)**
		30th April,2025(for January session)

This assignment has two questions. Answer both the questions. These questions carry 40 marks. Rest 10 marks are for viva-voce. Write C++ program and take its output as part of solution. Please go through the guidelines regarding the assignments given in the Programme Guide for the format of presentation.

Q1. Write a program in C++ for addition of two sparse Polynomials using Pointers. **(20 Marks)**

Q2. Write a program in C++ to generate ranks for the candidates based on the marks secured by them in an entrance examination. Make necessary assumptions. **(10 Marks)**

Q3. Write a program in C++ to create a book of 10 input pages. Make necessary assumptions. **(10 Marks)**

Q.N.1. Write a program in C++ for addition of two sparse Polynomials using Pointers

⟶

Program :

```
#include <iostream>
using namespace std;

// Structure to represent a node in the linked list
struct Node {
    int coeff;
    int pow;
    Node* next;
};

// Function to create a new node
Node* createNode(int coeff, int pow) {
    Node* newNode = new Node();
    newNode->coeff = coeff;
    newNode->pow = pow;
    newNode->next = nullptr;
    return newNode;
}

// Function to add a new node to the end of the linked list
void appendNode(Node*& head, int coeff, int pow) {
    Node* newNode = createNode(coeff, pow);
    if (head == nullptr) {
        head = newNode;
    } else {
        Node* temp = head;
        while (temp->next != nullptr) {
            temp = temp->next;
        }
        temp->next = newNode;
    }
}
```

```
// Function to add two polynomials
Node* addPolynomials(Node* poly1, Node* poly2) {
    Node* result = nullptr;
    while (poly1 != nullptr && poly2 != nullptr) {
        if (poly1->pow > poly2->pow) {
            appendNode(result, poly1->coeff, poly1->pow);
            poly1 = poly1->next;
        } else if (poly1->pow < poly2->pow) {
            appendNode(result, poly2->coeff, poly2->pow);
            poly2 = poly2->next;
        } else {
            int sumCoeff = poly1->coeff + poly2->coeff;
            appendNode(result, sumCoeff, poly1->pow);
            poly1 = poly1->next;
            poly2 = poly2->next;
        }
    }

    // If poly1 has remaining terms
    while (poly1 != nullptr) {
        appendNode(result, poly1->coeff, poly1->pow);
        poly1 = poly1->next;
    }

    // If poly2 has remaining terms
    while (poly2 != nullptr) {
        appendNode(result, poly2->coeff, poly2->pow);
        poly2 = poly2->next;
    }

    return result;
}

// Function to print the polynomial
void printPolynomial(Node* poly) {
    while (poly != nullptr) {
        cout << poly->coeff << "x^" << poly->pow;
        poly = poly->next;
        if (poly != nullptr) {
            cout << " + ";
        }
    }
    cout << endl;
}
```

```
int main() {
    Node* poly1 = nullptr;
    Node* poly2 = nullptr;

    // Creating first polynomial: 5x^3 + 4x^2 + 2x^0
    appendNode(poly1, 5, 3);
    appendNode(poly1, 4, 2);
    appendNode(poly1, 2, 0);

    // Creating second polynomial: 3x^3 + 4x^1 + 1x^0
    appendNode(poly2, 3, 3);
    appendNode(poly2, 4, 1);
    appendNode(poly2, 1, 0);

    // Printing the polynomials
    cout << "First Polynomial: ";
    printPolynomial(poly1);
    cout << "Second Polynomial: ";
    printPolynomial(poly2);

    // Adding the two polynomials
    Node* sum = addPolynomials(poly1, poly2);

    // Printing the sum
    cout << "Sum of Polynomials: ";
    printPolynomial(sum);

    return 0;
}
```

Explanation:

- **Node Structure:** This structure represents each term in the polynomial, containing the coefficient, power, and a pointer to the next node.
- **createNode:** Creates a new node with the given coefficient and power.
- **appendNode:** Adds a new node at the end of the linked list.
- **addPolynomials**: This function takes two linked lists (representing polynomials) and adds them. It handles cases where the exponents are the same or different.
- **printPolynomial:** A utility function to display the polynomial in a readable format.

Example Input:

1. First Polynomial: `5x^3 + 4x^2 + 2`
2. Second Polynomial: `3x^3 + 4x + 1`

Sample Output:

```
First Polynomial: 5x^3 + 4x^2 + 2x^0
Second Polynomial: 3x^3 + 4x^1 + 1x^0
Sum of Polynomials: 8x^3 + 4x^2 + 4x^1 + 3x^0
```

Detailed Explanation of Output:

- The first polynomial `5x^3 + 4x^2 + 2x^0` is added to the second polynomial `3x^3 + 4x^1 + 1x^0`.
- The terms with the same power are combined:
  - $5x^3 + 3x^3 = 8x^3$
  - $4x^2$ (no matching term in the second polynomial, so it remains as is)
  - $4x^1$ (only in the second polynomial)
  - $2x^0 + 1x^0 = 3x^0$
- The final sum polynomial is `8x^3 + 4x^2 + 4x^1 + 3x^0`.

Q2. Write a program in C++ to generate ranks for the candidates based on the marks secured by them in an entrance examination. Make necessary assumptions.

⟶

Assumptions:

- The number of candidates will be entered by the user.
- Marks will be integers, and candidates with the same marks will receive the same rank.

Program :

```
#include <iostream>
#include <algorithm>
using namespace std;

// Structure to represent a candidate with their marks and rank
struct Candidate {
    int id;
    int marks;
    int rank;
};

// Comparison function to sort candidates by marks in descending order
bool compare(Candidate a, Candidate b) {
    return a.marks > b.marks;
}

int main() {
    int n;

    // Input: Number of candidates
    cout << "Enter the number of candidates: ";
    cin >> n;

    Candidate candidates[n];

    // Input: Marks of candidates
    for (int i = 0; i < n; i++) {
        cout << "Enter marks for candidate " << i + 1 << ": ";
        candidates[i].id = i + 1;
        cin >> candidates[i].marks;
    }
```

```
    // Sort candidates by marks in descending order
    sort(candidates, candidates + n, compare);

    // Assign ranks
    for (int i = 0; i < n; i++) {
        if (i > 0 && candidates[i].marks == candidates[i - 1].marks) {
            candidates[i].rank = candidates[i - 1].rank; // Same rank for equal
marks
        } else {
            candidates[i].rank = i + 1; // Assign rank based on position
        }
    }

    // Output: Display the ranks of candidates
    cout << "\nCandidate ID\tMarks\tRank\n";
    for (int i = 0; i < n; i++) {
        cout << candidates[i].id << "\t\t" << candidates[i].marks << "\t" <<
candidates[i].rank << endl;
    }

    return 0;
}
```

Explanation:

- Structure: A Candidate structure is defined to hold the candidate's ID, marks, and rank.
- Sorting: The candidates are sorted in descending order of their marks using the sort function with a custom comparator.
- Rank Assignment: Ranks are assigned after sorting. If two candidates have the same marks, they are assigned the same rank.
- Input/Output: The program takes the number of candidates and their respective marks as input and prints the candidate ID, marks, and assigned rank as output.

Sample Input/Output:

Input:

```
Enter the number of candidates: 5
Enter marks for candidate 1: 85
Enter marks for candidate 2: 90
Enter marks for candidate 3: 75
Enter marks for candidate 4: 90
Enter marks for candidate 5: 65
```

Output:

```
Candidate ID    Marks    Rank
2               90       1
4               90       1
1               85       3
3               75       4
5               65       5
```

Explanation of Output:

- Candidates 2 and 4 have the highest marks (90), so both are ranked 1st.
- Candidate 1 has the next highest marks (85), so they are ranked 3rd.
- Candidate 3 has marks of 75, so they are ranked 4th.
- Candidate 5 has the lowest marks (65), so they are ranked 5th.

Q3. Write a program in C++ to create a book of 10 input pages. Make necessary assumptions.

⟶

Assumptions:

- Each page will have a title and content.
- The program will take input for the title and content of 10 pages.
- The book will be displayed after all 10 pages are entered.

Program :

```
#include <iostream>
#include <string>
using namespace std;

// Structure to represent a page in the book
struct Page {
    string title;
    string content;
};

// Function to create a book of 10 pages
void createBook(Page book[], int numPages) {
    for (int i = 0; i < numPages; i++) {
        cout << "Enter the title of page " << i + 1 << ": ";
        getline(cin, book[i].title);  // Input the title of the page
        cout << "Enter the content of page " << i + 1 << ": ";
        getline(cin, book[i].content);  // Input the content of the page
        cout << endl;
    }
}

// Function to display the book
void displayBook(Page book[], int numPages) {
    cout << "Displaying the book with " << numPages << " pages:\n\n";
    for (int i = 0; i < numPages; i++) {
        cout << "Page " << i + 1 << ": " << book[i].title << endl;
        cout << "Content: " << book[i].content << endl;
        cout << "----------------------\n";
    }
}
```

```
int main() {
    const int numPages = 10;
    Page book[numPages];

    // Create the book by taking input for 10 pages
    createBook(book, numPages);

    // Display the complete book
    displayBook(book, numPages);

    return 0;
}
```

Explanation:

- **Structure:** A Page structure is created to represent a page in the book. Each page has a title and content.
- **createBook Function:** This function takes input from the user for the title and content of each page.
- **displayBook Function:** This function displays the entire book, showing each page's title and content.
- **Main Function:** It initializes a book with 10 pages, calls the createBook function to take input, and then calls displayBook to show the content of the book.

Sample Input/Output:

Input:

```
Enter the title of page 1: Introduction
Enter the content of page 1: This is the introduction to the book.

Enter the title of page 2: Chapter 1
Enter the content of page 2: The content of chapter 1 begins here.

Enter the title of page 3: Chapter 2
Enter the content of page 3: This is the second chapter of the book.

... (input for remaining 7 pages)
```

Output:

```
Displaying the book with 10 pages:

Page 1: Introduction
Content: This is the introduction to the book.
----------------------

Page 2: Chapter 1
Content: The content of chapter 1 begins here.
----------------------

Page 3: Chapter 2
Content: This is the second chapter of the book.
----------------------

... (display for remaining 7 pages)
```

Explanation of Output:

- The program allows the user to input the title and content for each page.
- After all 10 pages are entered, the complete book is displayed with proper formatting.