National University of Computer and Emerging Sciences

Lab Manual
Data Structure

Course Instructor	Mr. Razi ulldin
Lab Instructor (s)	Ms. Mamoona Akbar Mr. Sukhan Amir
Section	BSE 3A
Semester	FALL 2023

Department of Computer Science
FAST-NU, Lahore, Pakistan

Objectives

After performing this lab, students shall be able to:

- POINTERS
- OOP concept

Task 1: Pointers

A ragged array is an array which contains a varying number of elements in each row. The Pascal triangle can be used to compute the coefficients of the terms in the expansion of (a + b) n.

Write a function that creates a ragged array representing the Pascal triangle. In a Pascal triangle, each element is the sum of the element directly above it (if any) and the element to the left of the element directly above it (if any).

A Pascal triangle of size 7 is shown below:

```
1
1 1
1 2 1
1 3 3 1
1 4 6 4 1
1 5 10 10 5 1
1 6 15 20 15 6 1
```

You are required to define the following functions:

int createPascalTriangle (int n) :** This function will take an integer (n) as argument and create a Pascal triangle consisting of n rows. It will dynamically allocate the two-dimensional ragged array, fill up its elements, and return a pointer to this filled array (Pascal triangle).

void displayPascalTriangle (int pt, int n) :** This function will take a pointer pt which is pointing to a Pascal triangle consisting of n rows. It will display the Pascal triangle on screen.

void deallocatePascalTriangle (int pt, int n):** This function will take a pointer pt which is pointing to a Pascal triangle consisting of n rows. This function will deallocate the two-dimensional array containing the Pascal triangle.

Write a main function which asks the user to specify the value of n. After that the main should call the above functions to create a Pascal triangle of size n, display it on screen, and, finally, de-allocate all the dynamically allocated memory.

Task 2: Association

Implement two classes:
● Student
● Course.
There is an association between them, where a Course can have multiple Student objects enrolled in it.
Class student has two data members.
● char* name
● Int studentid
And has following member functions.
● Student(const char*& name, int studentId) : name(name), studentId(studentId);
● char* getName() const;
● int getStudentId() const

Class Course has following data members
● string courseName
● Student* students
And has following members functions.
● Course(const std::char*& courseName) : courseName(courseName)
● void addStudent(Student* student)
● void displayStudents() const

```
int main()
{
Student student1("John Smith", 12345);
Student student2("Jane Doe", 54321);
Course mathCourse("Mathematics");
mathCourse.addStudent(&student1);
mathCourse.addStudent(&student2);
mathCourse.displayStudents();
return 0;
}
```

Task 3: Inheritance

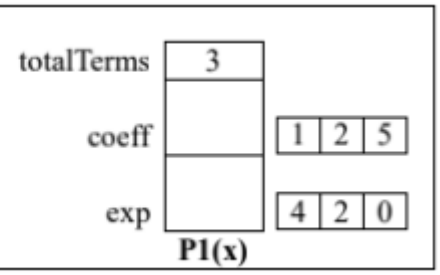
- Define a base class "LibraryItem" that represents a generic item available in the library, such as a book, magazine, or DVD. Include attributes like "item_id," "title," "author/creator", "publication_year," and "is_available" (to track item availability), along with relevant methods.
- Create two derived classes "Book" and "Magazine" that inherit from the "LibraryItem" class. Add additional attributes specific to books and magazines, such as "ISBN" and "genre" for books, and "issue_number" and "topic" for magazines.
- Implement a function to add new books and magazines to the library system, displaying their details like title, author/creator, and other specific attributes.
- Create a derived class "DVD" that also inherits from the "LibraryItem" class. Add attributes such as "duration" and "director" for DVDs.
- Implement a function to add new DVDs to the library system, displaying their details like title, director, and duration.
- Write a function that allows users to borrow library items and update their availability status.
- Implement functions to display a list of all library items, books, magazines, and DVDs separately.
- Create objects of each class and demonstrate the functionalities of the library management system, including adding items, borrowing items, and displaying lists.

- Test the program by adding various library items, borrowing some of them, and displaying the available items.

Task 4: Operator Overloading

A polynomial P1(x) = x4+2x2+5 has three terms: x^4, $2x^2$ and 5. Coefficients of these terms are 1, 2 and 5 respectively while exponents are 4, 2 and 0 respectively. To work with Polynomials, a definition of class Polynomial is given below and memory configuration for P1 is shown as follows:

```
class Polynomial
{
private:
int totalTerms;//Total terms in a Polynomial
int* coeff;//to save array of coefficients
int* exp; //to save array of exponents
};
```



Your task is to complete the definition of Polynomial class such that the main program runs successfully. Make sure that your program doesn't consume extra memory space and it should not leak any memory.

```
void main()
{
int coeff_P1[] = {1,2,5}; //Coefficients for Polynomial P1
int exp_P1[] = {4,2,0}; //Exponents for Polynomial P1
int coeff_P2[] = {4,3}; //Coefficients for Polynomial P2
int exp_P2[] = {6,2}; //Exponents for Polynomial P2
Polynomial P1(3, coeff_P1, exp_P1);//Creates P1 with 3 terms (P1 = 1x^4 + 2x^2 +
5x^0 )
Polynomial P2(2, coeff_P2, exp_P2);//Creates P2 with 2 terms (P2 = 4x^6 + 3x^2)
cout<<"P1 = "<<P1<<endl; //Prints P1 = x^4+2x^2+5
cout<<"P2 = "<<P2<<endl; //Prints P2 = 4x^6+3x^2
if(!P1)
cout<<"P1 is zero"<<endl; /*if the polynomial has only 1 term and its coeff and exp are zero. i.e.
if p1 = 0.*/
if(P1 != P2)
cout<<"P1 is Not Equal to P2"<<endl;
cout<<++P1<<endl; //adds 1 in all the coefficients.
cout<<P1<<endl;
cout<<P1++<<endl; //adds 1 in all the coefficients.
cout<<P1<<endl;
Polynomial P3 = P1+P2; //Adds P1 and P2 and saves result in P3.You may
consume extra space for resultant Polynomial in Add function
cout<<"P3 = "<<P3<<endl; //Prints P3 = 4x^6+x^4+5x^2+5
P3 = P1+2;
cout<<"P3 = "<<P3<<endl;

P4 = P1;
cout<<"P4 = "<<P4<<endl;
}
```