Graphic Era HILL UNIVERSITY

Established by an Act of the State Legislature of Uttarakhand (Adhiniyam Sankhya 12 of 2011)

Practical File

OOPS with C++ Lab (PCS 307)

B. Tech Third Semester
Session : 2024 - 25

Submitted to:
Dr. Shilpa Jain
Assistant Professor
Computer Science & Engineering
Graphic Era Hill University
Bhimtal Campus

Submitted by:
Khajan Bhatt
B. Tech (CSE)
Section A
Roll No- 2361xxx

COLLEGE ROLL NO : ________   EXAMINATION ROLL NO : ______

Graphic Era HILL UNIVERSITY
Established by an Act of the State Legislature of Uttarakhand (Adhiniyam Sankhya 12 of 2011)

THIS IS TO CERTIFY THAT **MR./MS. ________________________** HAS SATISFACTORILY COMPLETED ALL THE EXPERIMENTS IN THE LABORATORY OF THIS COLLEGE. THE COURSE OF THE EXPERIMENTS / TERM WORK IN **OOPS WITH C++ LAB (PCS 307)** IN PARTIAL FULLMENT OF THE REQUIREMENT IN **THIRD SEMESTER** OF **BACHELOR OF TECHNOLOGY (C.S.E.)** DEGREE COURSE PRESCRIBED BY THE GRAPHIC ERA HILL UNIVERSITY, BHIMTAL DURING THE YEAR **2024 - 25**.

CONCERNED FACULTY   HEAD OF DEPARTMENT

NAME OF EXAMINER :

SIGNATURE OF EXAMINER :

Index

Index

6		Define a class Hotel in C++ with the following specifications: **Private members:** • Rno : Data member to store room number • Name : Data member to store customer name • Tariff : Data member to store per day charges • NOD : Data member to store number of days of stay • CALC() : Function to calculate and return amount as NOD*Tariff ,and if the value of days* Tariff >10000, then total amount is 1.05* days*Tariff. **Public members:** • Checkin() Function to enter the content Rno, Name, Tariff and NOD • Checkout() Function to display Rno, Name, Tariff, NOD and Amount (amount to be displayed by calling function) CALC()	9 - 10	
7		Implement a Program in C++ by defining a class to represent a bank account. Include the following: **Data Member:** Name of the depositor, Account number, Type of account (Saving, Current etc.), Balance amount in the account. **Member Functions:** To assign initial values, To deposit an amount, To withdraw an amount after checking the balance, To display name and balance.	11 - 12	
8		Anna is a contender for valedictorian of her high school. She wants to know how many students (if any) have scored higher than her in the exams given during this semester. Create a class named Student with the following specifications: • An instance variable named scores holds a student's 5 exam scores.	13 - 14	

Index

9		• A void input () function reads 5 integers and saves them to scores. • An int calculateTotalScore() function that returns the sum of the student's scores.		
9		Construct a Program in C++ to show the working of function overloading(compile time polymorphism) by using a function named calculate Area () to calculate area of square, rectangle and triangle using different signatures as required.	15	
10		Create a class called Invoice that a hardware store might use to represent an invoice for an item sold at the store. An Invoice should include four pieces of information as instance. **Data Members:** • partNumber (type String) • partDescription (type String) • quantity of the item being purchased (type int) • price_per_item (type double) Your class should have a constructor that initializes the four instance variables. Provide a set and a get method for each instance variable. In addition, provide a method named getInvoiceAmount() that calculates the invoice amount (i.e., multiplies the quantity by the price per item), then returns the amount as a double value. If the quantity is not positive, it should be set to 0. If the price per item is not positive, it should be set to 0.0. Write a test application named invoiceTest that demonstrates class Invoice’s capabilities.	16 - 18	

PROGRAM OBJECTIVE

1. An electricity board charges the following rates to domestic users to discourage large consumption of energy.

- For the first 100 units :- 60 P per unit
- For the next 200 units :- 80 P per unit
- Beyond 300 units :- 90 P per unit

All users are charged a minimum of Rs 50. If the total amount is more than Rs 300, then an additional surcharge of 15% is added.

Implement a C++ program to read the names of users and number of units consumed and display the charges with names.

SOLUTION

```
#include <iostream>
using namespace std;
int main(){
	int units;
	double totalCharges {0};
	cout<<"\nEnter number of units consumed : "; cin>>units;
	if (units <= 100)
		totalCharges = units * 0.60;
	else if (units <= 300)
		totalCharges = (100 * 0.60) + ((units - 100) * 0.80);
	else
		totalCharges = (100 * 0.60) + (200 * 0.80) + ((units - 300) *
	0.90);
	if (totalCharges < 50) { totalCharges = 50;}
	if (totalCharges > 300){ totalCharges += totalCharges * 0.15;}
	cout << "Units Consumed : "<< units << endl;
	cout << "Total Charges : Rs. " << totalCharges << endl;
	return 0;
}
```

/***OUTPUT:**

```
Enter number of units consumed : 8923
Units Consumed : 8923
Total Charges : Rs. 9177.81*/
```

PROGRAM OBJECTIVE

2. Construct a C++ program that removes a specific character from a given string and return the updated string.

Typical Input: computer science is the future
Typical Output: compuer science is he fuure

SOLUTION

```
#include <iostream>
using namespace std;
string removeCharacter (string & inputStr, char charToRemove){
     string resultStr;
     for (const char & i : inputStr){
          if (i != charToRemove)
               resultStr += i;
     }
     return resultStr;
}
int main(){
     string inputStr; char charToRemove;
     cout<<"\nEnter the string : ";
     getline(cin, inputStr);
     cout<<"Enter the character to be removed : ";
     cin>>charToRemove;
     string resultStr = removeCharacter(inputStr, charToRemove);
     cout<<"Modified String : "<<resultStr<<endl;
     return 0;
}
```

/***OUTPUT:**

```
Enter the string : graphic era hill university
Enter the character to be removed : i
Modified String : graphc era hll unversty */
```

PROGRAM OBJECTIVE

3. Implement a C++ program to find the non-repeating characters in string.

Typical Input: graphic era university
Typical Output: c g h n p s t u v y

SOLUTION

```
#include <iostream>
using namespace std;
//Brute Force Approach - Time : O(n²), Space : O(1)
void printCharacter (string & inputStr){
     cout<<"Non-Repeating Characters : ";
     for (int i = 0; inputStr[i] != '\0'; i++){
          char curChar = inputStr[i];
          bool doPrint = true;
          for (int j = 0; inputStr[j] != '\0'; j++){
               if(curChar == inputStr[j] && i != j) {
                    doPrint = false;
                    break;
               }
          }
          if (doPrint == true) cout<<curChar<<" ";
     }
     cout<<endl;
     return;
}
int main(){
     string inputStr;
     cout<<"\nEnter the string : ";
     getline(cin, inputStr);
     printCharacter(inputStr);
     return 0;
}
```

/***OUTPUT:**
```
Enter the string : graphic era university
Non-Repeating Characters : g p h c u n v s t y */
```

OPTIMIZATIONS

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

//Hashing Approach - Time : O(n), Space : O(n)
void printCharacter (string & inputStr){
     cout<<"Non-Repeating Characters : ";
     map <char, int> hash;
     for (const char & i : inputStr) hash[i]++;
     for (const pair <char, int> & i : hash)
          if (i.second == 1)
               cout<<i.first<<" ";
     cout<<endl;
}
int main(){
     string inputStr;
     cout<<"\nEnter the string : ";
     getline(cin, inputStr);
     printCharacter(inputStr);
     return 0;
}
```

/***OUTPUT:**

```
Enter the string : graphic era hill university
Non-Repeating Characters : c g n p s t u v y */
```

PROGRAM OBJECTIVE

4. You are given an array of elements. Now you need to choose the best index of this array. An index of the array is called best if the special sum of this index is maximum across the special sum of all the other indices.

To calculate the special sum for any index you pick the first element that is and add it to your sum. Now you pick next two elements i.e., and add both of them to your sum. Now you will pick the next elements, and this continues till the index for which it is possible to pick the elements.

Find the best index and in the output print its corresponding special sum. Note that there may be more than one best index, but you need to only print the maximum special sum.

Input: First line contains an integer as input. Next line contains space separated integers denoting the elements of the array.
Output: In the output you have to print an integer that denotes the maximum special sum.

Input/Output Format	
Typical Input	Typical Output
5 1 3 1 2 5	8
10 2 1 3 9 2 4 -10 -9 1 3	9

SOLUTION

```
#include <iostream>
using namespace std;
//Brute Force Approach - Time : O(n³), Space : O(n)
int specialSum (int * nums, int size){
    int temp {INT_MIN};
    for(int i = 0; i < size; i++){
        int sum = 0;
        for(int k = i, y = 1; k + y <= size; k += y, y++){
            for (int l = 0; l < y; l++){ sum += nums[k + l]; }
        }
        if(temp<sum){ temp = sum; }
    }
    return temp;
}
int main(){
    int size;
    cout<<"\nEnter number of Elements : "; cin>>size;
    int nums [size];
    cout<<"Enter the numbers : ";
    for (int i = 0; i < size; i++) cin>>nums[i];
    cout<<"Special Sum : "<<specialSum(nums, size)<<endl
    return 0;
}
```

```
/*OUTPUT:
Enter number of Elements : 10
Enter the numbers : 2 1 3 9 2 4 -10 -9 1 3
Special Sum : 9*/
```

OPTIMIZATIONS

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

//Inner Summation Approach - Time : O(n²), Space - O(1)
int specialSum_1 (int * nums, int size){
    int maxSpecialSum {INT_MIN};
    for (int i = 0; i < size; i++){
        int specialSum{0}, k = 1, n = 1;
        for (int j = i; i + k <= size; j++){
            specialSum += nums[j];
            if (j - i == k - 1) {++n; k = (n * (n + 1))/2;}
        }
        if (specialSum > maxSpecialSum) maxSpecialSum = specialSum;
    }
    return maxSpecialSum;
}


//Hashing Approach - Time : O(n), Space : O(n)
int specialSum_2 (int * nums, int size){
    int maxSpecialSum {0};
    vector <int> hash; hash.push_back(0);
    for (int i = 0; i < size; i++){
        maxSpecialSum += nums[i];
        hash.push_back(maxSpecialSum);
    }
    maxSpecialSum = INT_MIN;
    for (int i = 0; i < size; i++){
        int n = (-1 + static_cast <int> (sqrt(8*(size - i) + 1)))/2;
        int j = (n * (n + 1))/2 + i;
        int specialSum = hash[j] - hash[i];
        if (specialSum > maxSpecialSum) maxSpecialSum = specialSum;
    }
    return maxSpecialSum;
}

int main(){
    int size;
    cout<<"\nEnter number of Elements : "; cin>>size;
    int nums [size];
    cout<<"Enter the numbers : ";
    for (int i = 0; i < size; i++) cin>>nums[i];
    cout<<"\nSpecial Sum using 1 : "<<specialSum_1(nums, size)<<endl
        <<"Special Sum using 2 : "<<specialSum_2(nums, size)<<endl;
    return 0;
}
```

/***OUTPUT:**

```
Enter number of Elements : 10
Enter the numbers : 2 1 3 9 2 4 -10 -9 1 3
Special Sum using 1 : 9
Special Sum using 2 : 9*/
```

PROGRAM OBJECTIVE

5. Implement a C++ program to demonstrate the concept of data abstraction using the concept of Class and Objects.

SOLUTION

```
#include <iostream>
using namespace std;
class Employee {
        // Private Data members - Hidden from outside
        string name;
        double salary;
        static int id;
        static void giveId (void){ id++; } //Static member ID is abstracted
    public:
        // Public methods - Provides interface to the outside
        void setDetails(string empName, double empSalary) {
            name = empName;
            salary = empSalary;
            giveId();
        }
        void displayDetails() {
            cout << "\nEmployee Name : " << name << endl;
            cout << "Employee ID : " << id << endl;
            cout << "Employee Salary : $" << salary << endl;
        }
};
int Employee :: id = 0;
int main() {
    Employee emp;
    emp.setDetails("Khajan Bhatt", 55000.50);
    emp.displayDetails();
    return 0;
}
```

/***OUTPUT:**

```
Employee Name : Khajan Bhatt
Employee ID : 1
Employee Salary : 55000.5 */
```

PROGRAM OBJECTIVE

6. Define a class Hotel in C++ with the following specifications:

Private members:

- Rno : Data member to store room number
- Name : Data member to store customer name
- Tariff : Data member to store per day charges
- NOD : Data member to store number of days of stay
- CALC() : Function to calculate and return amount as NOD*Tariff ,and if the value of days* Tariff >10000, then total amount is 1.05* days*Tariff.

Public members:

- Checkin() : Function to enter the content Rno, Name, Tariff and NOD
- Checkout() : Function to display Rno, Name, Tariff, NOD and Amount (amount to be displayed by calling function) CALC()

SOLUTION

```
#include <iostream>
using namespace std;
class Hotel {
    private:
        int RNo, NOD;
        string Name;
        float tarrif;
        float Calc (void){
            float totalAmount = NOD * tarrif;
            if (totalAmount > 10000) totalAmount *= 1.05;
            return totalAmount;
        }
    public:
        void CheckIn (void){
            cout<<"\nEnter Room Number : "; cin>>RNo;
            cin.ignore(INT_MAX, '\n');
            cout<<"Enter Customer Name : "; getline(cin, Name);
            cout<<"Enter Tarrif per Day : "; cin>>tarrif;
            cout<<"Enter the Number of Days of stay : "; cin>>NOD;
            return;
        }
        void CheckOut (void){
            cout<<"\nDetails : "<<endl<<endl
```

```
                <<"\tRoom Number : "<<RNo<<endl
                <<"\tName of Customer : "<<Name<<endl
                <<"\tTarrif per Day : "<<tarrif<<endl
                <<"\tNumber of Days of stay : "<<NOD<<endl<<endl
                <<"Total amount to Pay : "<< Calc()<<endl;
            return;
        }
};
int main(){
    Hotel royalWindsor;
    royalWindsor.CheckIn();
    royalWindsor.CheckOut();
    return 0;
}
```

/***OUTPUT:**

```
Enter Room Number : 68
Enter Customer Name : Khajan Bhatt
Enter Tarrif per Day : 5500
Enter the Number of Days of stay : 3

Details :

        Room Number : 68
        Name of Customer : Khajan Bhatt
        Tarrif per Day : 5500
        Number of Days of stay : 3

Total amount to Pay : 17325*/
```

PROGRAM OBJECTIVE

7. Implement a Program in C++ by defining a class to represent a bank account. Include the following:

Data Member:

- Name of the depositor
- Account number
- Type of account (Saving, Current etc.)
- Balance amount in the account.

Member Functions:

- To assign initial values
- To deposit an amount
- To withdraw an amount after checking the balance
- To display name and balance.

SOLUTION

```
#include <iostream>
using namespace std;
class BankAccount {
    private:
        long long accNo;
        string Name, accType;
        float balance;
    public:
        BankAccount(string Name, int accNo, string accType, float balance){
            this->Name = Name;
            this->accNo = accNo;
            this->accType = accType;
            this->balance = balance;
        }
        void deposit (float amount){
            if (amount > 0) {
                balance += amount;
                cout << "\nSuccessfully deposited : " << amount;
            }
            else cout << "\nInvalid deposit amount!";
        }
        void withdraw(float amount) {
            if (amount > 0) {
                if (amount <= balance) {
                    balance -= amount;
                    cout << "\nSuccessfully withdrawn : " << amount;
                }
                else cout << "\nInsufficient balance for withdrawal!";
            }
            else cout << "\nInvalid withdrawal amount!";
        }
        void display() {
            cout<<"\n\nDepositor Details : "<<endl
                <<"\tDepositor Name : "<<Name<<endl
                <<"\tAccount Number : "<<accNo<<endl
                <<"\tAccount Type : "<<accType<<endl
                <<"\tCurrent Balance : "<<balance<<endl;
        }
};
int main(){
    BankAccount bankAccount("Khajan Bhatt", 123456789, "Current", 10000.0);
    bankAccount.display();
    bankAccount.deposit(1500.0);
    bankAccount.withdraw(500.0);
    bankAccount.display();
    bankAccount.withdraw(16000.0);
    return 0;
}
```

/***OUTPUT:**

```
Depositor Details :
    Depositor Name : Khajan Bhatt
    Account Number : 123456789
    Account Type : Current
    Current Balance : 10000

Successfully deposited : 1500
Successfully withdrawn : 500

Depositor Details :
    Depositor Name : Khajan Bhatt
    Account Number : 123456789
    Account Type : Current
    Current Balance : 11000

Insufficient balance for withdrawal!*/
```

PROGRAM OBJECTIVE

8. Anna is a contender for valedictorian of her high school. She wants to know how many students (if any) have scored higher than her in the exams given during this semester. Create a class named Student with the following specifications:

- An instance variable named scores holds a student's 5 exam scores.
- A void input () function reads 5 integers and saves them to scores.

An int calculateTotalScore() function that returns the sum of the student's scores.

Input Format:

In the void Student::input() function, you must read 5 scores from standard input and save them to your scores instance variable.

Output Format:

In the int Student::calculateTotalScore() function, you must return the student's total grade (the sum of the values in scores). The code in the editor will determine how many scores are larger than Anna's and print that number to the console.

Sample Input:

The first line contains n, the number of students in Anna's class. The n subsequent lines contain each student's 5 exam grades for this semester.

```
3
30 40 45 10 10
40 40 40 10 10
50 20 30 10 10
```

Sample Output:

```
1
```

SOLUTION

```
#include <iostream>
using namespace std;
class Student {
    private:
        int scores [5];
    public:
        void input() {
            for (int i = 0; i < 5; ++i)
                cin >> scores[i];
        }
        int calculateTotalScore() {
            int totalScore = 0;
            for (int i = 0; i < 5; ++i)
                totalScore += scores[i];
            return totalScore;
        }
};
int main() {
    int n;
    cout<<"\nEnter number of students : ";
    cin >> n;
    Student students[n];
    cout<<"\nEnter Marks in different Subjects : "<<endl<<endl;
    for (int i = 0; i < n; ++i){
        cout<<"Student "<<i + 1<<" : ";
        students[i].input();
    }
    int annaTotalScore = students[0].calculateTotalScore();
    int countHigher = 0;
    for (int i = 1; i < n; ++i){
        if (students[i].calculateTotalScore() > annaTotalScore) {
countHigher++; }
    }
    cout<<"\nNumber of students having better grades than Anna :
"<<countHigher << endl;
    return 0;
}
```

/***OUTPUT:**

```
Enter number of students : 3

Enter Marks in different Subjects :

Student 1 : 30 40 45 10 10
Student 2 : 40 40 40 10 10
Student 3 : 50 20 30 10 10

Number of students having better grades than Anna : 1 */
```

PROGRAM OBJECTIVE

9. Construct a Program in C++ to show the working of function overloading (compile time polymorphism) by using a function named calculate Area () to calculate area of square, rectangle and triangle using different signatures as required.

SOLUTION

```
#include <iostream>
using namespace std;
int calculateArea(int side) {
    return side * side;
}
int calculateArea(int length, int breadth) {
    return length * breadth;
}
double calculateArea(double base, double height) {
    return 0.5 * base * height;
}
int main() {
    int squareArea = calculateArea(5);
    cout << "Area of Square : " << squareArea << endl;
    int rectangleArea = calculateArea(4, 6);
    cout << "Area of Rectangle : " << rectangleArea << endl;
    double triangleArea = calculateArea(3.0, 5.0);
    cout << "Area of Triangle : " << triangleArea << endl;
    return 0;
}
```

/***OUTPUT:**

```
Area of Square : 25
Area of Rectangle : 24
Area of Triangle : 7.5*/
```

PROGRAM OBJECTIVE

10. Create a class called Invoice that a hardware store might use to represent an invoice for an item sold at the store. An Invoice should include four pieces of information as instance.

Data Members:

- partNumber (type String)
- partDescription (type String)
- quantity of the item being purchased (type int)
- price_per_item (type double)

Your class should have a constructor that initializes the four instance variables. Provide a set and a get method for each instance variable.

In addition, provide a method named getInvoiceAmount() that calculates the invoice amount (i.e., multiplies the quantity by the price per item), then returns the amount as a double value. If the quantity is not positive, it should be set to 0. If the price per item is not positive, it should be set to 0.0.

Write a test application named invoiceTest that demonstrates class Invoice's capabilities.

SOLUTION

```
#include <iostream>
#include <string>
using namespace std;
class Invoice {
    private:
        string partNumber;
        string partDescription;
        int quantity;
        double price_per_item;
    public:
        Invoice(string partNum, string partDesc, int qty, double price) {
            partNumber = partNum;
            partDescription = partDesc;
            setQuantity(qty);
            setPricePerItem(price);
        }
        void setPartNumber(string partNum) {
            partNumber = partNum;
        }
        string getPartNumber() const {
            return partNumber;
        }
        void setPartDescription(string partDesc) {
            partDescription = partDesc;
        }
        string getPartDescription() const {
            return partDescription;
        }
        void setQuantity(int qty) {
            if (qty < 0) { quantity = 0; }
            else { quantity = qty;}
        }
        int getQuantity() const {
            return quantity;
        }
        void setPricePerItem(double price) {
            if (price < 0) { price_per_item = 0.0; }
            else { price_per_item = price; }
        }
        double getPricePerItem() const {
            return price_per_item;
        }
        double getInvoiceAmount() const {
            return quantity * price_per_item;
        }
};

int main() {
    Invoice invoice1("1234", "Hammer", 10, 15.50);

    cout << "Part Number: " << invoice1.getPartNumber() << endl;
    cout << "Part Description: " << invoice1.getPartDescription() << endl;
    cout << "Quantity: " << invoice1.getQuantity() << endl;
    cout << "Price per Item: $" << invoice1.getPricePerItem() << endl;
    cout << "Total Invoice Amount: $" << invoice1.getInvoiceAmount() <<
endl;

    invoice1.setQuantity(-5);
    invoice1.setPricePerItem(-20.0);

    cout << "\nAfter setting invalid values (negative quantity and price):"
<< endl;
    cout << "Quantity: " << invoice1.getQuantity() << endl;
    cout << "Price per Item: $" << invoice1.getPricePerItem() << endl;
    cout << "Total Invoice Amount: $" << invoice1.getInvoiceAmount() <<
endl;
    return 0;
}
```

/***OUTPUT:**

```
Part Number: 1234
Part Description: Hammer
Quantity: 10
Price per Item: $15.5
Total Invoice Amount: $155

After setting invalid values (negative quantity and price):
Quantity: 0
Price per Item: $0
Total Invoice Amount: $0*/
```