**Programs-7 Write a program to input a string (can be a sentence). Convert to capital and add 2 in each character present in the string. Print input and modified string.**

**Sample input: Adc MOP Boy Girl XYZ LUX**

**Sample Output: CFE OQR DQA IKTN ZAP NWZ**

void capitalizeAndAdd2(string str){

string original = str;

str = str.ToUpper();

string newStr = "";

for (int i = 0; i < str.Length; i++){

char ch = str[i];

if(ch != ' '){

ch = (char)(ch + 2);

if(ch > 'Z'){

ch = (char)((ch % 91) + 'A');

}

};

newStr += ch;

}

Console.WriteLine($"Input: {original}");

Console.WriteLine($"Output: {newStr}");

}

**Output-**



**Program-8 Write a program to input a string (can be a sentence).Invert the case of the string. Print input and modified string.**

**Sample input: JaMes BonD**

**Sample ouput: jAmES bONd**

void invertCase(string str)

{

char[] charArr = str.ToUpper().ToCharArray();

string newStr = "";

for (int i = 0; i < str.Length; i++)

{

if (charArr[i] != ' ' && charArr[i] == str[i])

{

charArr[i] = (char)(charArr[i] + 32);

}

newStr += charArr[i];

}

Console.WriteLine($"Input: {str}");

Console.WriteLine($"Output: {newStr}");

}

**Ouput-**



**Program-9 Write a program to accept a word, convert it to lowercase if it is in uppercase, and display the new word by replacing the vowels with the character following it.**

void vowelModifier(string str)

{

char[] charArr = str.ToLower().ToCharArray();

string newStr = "";

for (int i = 0; i < str.Length; i++)

{

if (charArr[i] != ' ' && "aeiou".Contains(charArr[i]))

{

charArr[i] = (char)(charArr[i] + 1);

}

newStr += charArr[i];

}

Console.WriteLine($"Input: {str}");

Console.WriteLine($"Output: {newStr}");

}

**Ouput-**



**Program-10 Create class “ArrangeLetters” and a method arrange with a string param. Convert word into capitals. Arrange each letter of word in alphabetic order. Print word before and after arranging letters in A-Z order**

**Sample input: BaSica**

**Output:**

**Word in capital: BASICA**

**Word after sorting: AABCIS**

class ArrangeLetters

{

public static void arrange(string str)

{

str = str.ToUpper();

char[] charArr = str.ToCharArray();

string newStr = "";

//Selection sort for the win !!!

for (int i = 0; i < charArr.Length - 1; i++)

{

for (int j = i + 1; j < charArr.Length; j++)

{

if (charArr[i] > charArr[j])

{

char temp = charArr[i];

charArr[i] = charArr[j];

charArr[j] = temp;

}

}

newStr += charArr[i];

}

newStr += charArr[charArr.Length - 1];

Console.WriteLine($"Word in Capital: {str}");

Console.WriteLine($"Word after sorting: {newStr}");

}

}

**Output-**



**Program-12 Write a program to create a banking system with three classes - Bank, Account, SavingsAccount, and CurrentAccount. The bank should have a list of accounts and methods for adding them. Accounts should be an interface with methods to deposit, withdraw, calculate interest, and view balances. SavingsAccount and CurrentAccount should implement the Account interface and have their own unique methods.**

using System;

using System.Collections.Generic;

namespace BankingSystem{

// Interface for Account

public interface IAccount{

string Id { get; set; }

string Name { get; set; }

double Balance { get; set; }

bool Deposit(double amount);

bool Withdraw(double amount);

double Interest();

void ViewBalance();

void Home();

}

// Saving Account Class

public class SavingAccount : IAccount{

public string Id { get; set; }

public string Name { get; set; }

public double Balance { get; set; }

public static double InterestRate = 2.0;

public static double MinBalance = 1000;

public SavingAccount(string id, string name, double openingBalance){

Id = id;

Name = name;

Balance = openingBalance;

}

public bool Deposit(double amount){

if (amount <= 0) return false;

Balance += amount;

return true;

}

public bool Withdraw(double amount){

if (amount <= 0 || Balance - amount < MinBalance) return false;

Balance -= amount;

return true;

}

public double Interest(){

return Balance \* InterestRate / 100;

}

public void ViewBalance(){

Console.WriteLine($"Your current balance is: {Balance}");

}

public void Home(){

Console.WriteLine($"Welcome to {Name}'s Savings Account");

while (true){

Console.WriteLine("Press 1 to Deposit");

Console.WriteLine("Press 2 to Withdraw");

Console.WriteLine("Press 3 to View Balance and Interest");

Console.WriteLine("Press 4 to Exit");

int choice = int.Parse(Console.ReadLine());

bool exit = false;

switch (choice){

case 1:

Console.WriteLine("Enter the amount to deposit:");

double depositAmount = double.Parse(Console.ReadLine());

if (Deposit(depositAmount)){

Console.WriteLine($"Deposit Successful! Your new balance is: {Balance}");

}

else{

Console.WriteLine("Invalid deposit amount");

}

break;

case 2:

Console.WriteLine("Enter the amount to withdraw:");

double withdrawAmount = double.Parse(Console.ReadLine());

if (Withdraw(withdrawAmount)){

Console.WriteLine($"Withdrawal Successful! Your new balance is: {Balance}");

}

else{

Console.WriteLine("Invalid withdrawal amount or insufficient balance.");

}

break;

case 3:

ViewBalance();

Console.WriteLine($"Your Interest is: {Interest()}");

break;

case 4:

Console.WriteLine($"Thank you {Name}");

exit = true;

break;

default:

Console.WriteLine("Invalid choice. Please try again.");

break;

}

if (exit) break;

}

}

}

// Current Account Class

public class CurrentAccount : IAccount{

public string Id { get; set; }

public string Name { get; set; }

public double Balance { get; set; }

public static double InterestRate = 1.0; // Current account usually gives no interest

public static double MinBalance = 10000;

public CurrentAccount(string id, string name, double openingBalance){

Id = id;

Name = name;

Balance = openingBalance;

}

public bool Deposit(double amount){

if (amount <= 0) return false;

Balance += amount;

return true;

}

public bool Withdraw(double amount){

if (amount <= 0 || Balance - amount < MinBalance) return false;

Balance -= amount;

return true;

}

public double Interest(){

Console.WriteLine("No Interest for Current Account Users");

return Balance \* InterestRate / 100;

}

public void ViewBalance(){

Console.WriteLine($"Your current balance is: {Balance}");

}

public void Home(){

Console.WriteLine($"Welcome to {Name}'s Current Account");

while (true){

Console.WriteLine("Press 1 to Deposit");

Console.WriteLine("Press 2 to Withdraw");

Console.WriteLine("Press 3 to View Balance and Interest");

Console.WriteLine("Press 4 to Exit");

int choice = int.Parse(Console.ReadLine());

bool exit = false;

switch (choice){

case 1:

Console.WriteLine("Enter the amount to deposit:");

double depositAmount = double.Parse(Console.ReadLine());

if (Deposit(depositAmount)){

Console.WriteLine($"Deposit Successful! Your new balance is: {Balance}");

}

else{

Console.WriteLine("Invalid deposit amount");

}

break;

case 2:

Console.WriteLine("Enter the amount to withdraw:");

double withdrawAmount = double.Parse(Console.ReadLine());

if (Withdraw(withdrawAmount)){

Console.WriteLine($"Withdrawal Successful! Your new balance is: {Balance}");

}

else{

Console.WriteLine("Invalid withdrawal amount or insufficient balance.");

}

break;

case 3:

ViewBalance();

Console.WriteLine($"Your Interest is: {Interest()}");

break;

case 4:

Console.WriteLine($"Thank you {Name}");

exit = true;

break;

default:

Console.WriteLine("Invalid choice. Please try again.");

break;

}

if (exit) break;

}

}

}

// Bank class to manage accounts

public class Bank{

private int globalId = 0;

private List<SavingAccount> savingAccounts = new List<SavingAccount>();

private List<CurrentAccount> currentAccounts = new List<CurrentAccount>();

// Add a new saving account

public void AddSavingAccount(string name, double openingBalance){

if (openingBalance < SavingAccount.MinBalance){

Console.WriteLine($"Minimum opening balance is {SavingAccount.MinBalance}");

return;

}

globalId++;

SavingAccount acc = new SavingAccount(globalId.ToString(), name, openingBalance);

savingAccounts.Add(acc);

Console.WriteLine($"Account added, your ID is {globalId}");

}

// Add a new current account

public void AddCurrentAccount(string name, double openingBalance){

if (openingBalance < CurrentAccount.MinBalance){

Console.WriteLine($"Minimum opening balance is {CurrentAccount.MinBalance}");

return;

}

globalId++;

CurrentAccount acc = new CurrentAccount(globalId.ToString(), name, openingBalance);

currentAccounts.Add(acc);

Console.WriteLine($"Account added, your ID is {globalId}");

}

// Go to an existing account

public void GotoAcc(string id){

foreach (var acc in savingAccounts){

if (acc.Id == id){

acc.Home();

return;

}

}

foreach (var acc in currentAccounts){

if (acc.Id == id){

acc.Home();

return;

}

}

Console.WriteLine("Account not found");

}

}

// Main Program

class Program{

static void Main(string[] args){

Bank bank = new Bank();

while (true){

Console.WriteLine("Welcome to Banking System");

Console.WriteLine("Press 1 to Add a new Saving Account");

Console.WriteLine("Press 2 to Add a new Current Account");

Console.WriteLine("Press 3 to Go to an Existing Account");

Console.WriteLine("Press 4 to Exit");

int choice = int.Parse(Console.ReadLine());

bool exit = false;

switch (choice){

case 1:

Console.WriteLine("Enter the name:");

string name = Console.ReadLine();

Console.WriteLine("Enter the opening balance:");

double openingBalance = double.Parse(Console.ReadLine());

bank.AddSavingAccount(name, openingBalance);

break;

case 2:

Console.WriteLine("Enter the name:");

string name2 = Console.ReadLine();

Console.WriteLine("Enter the opening balance:");

double openingBalance2 = double.Parse(Console.ReadLine());

bank.AddCurrentAccount(name2, openingBalance2);

break;

case 3:

Console.WriteLine("Enter the Account ID:");

string accountId = Console.ReadLine();

bank.GotoAcc(accountId);

break;

case 4:

Console.WriteLine("Thank you");

exit = true;

break;

default:

Console.WriteLine("Invalid choice. Please try again.");

break;

}

if (exit) break;

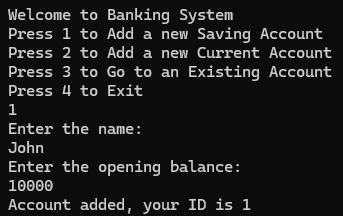
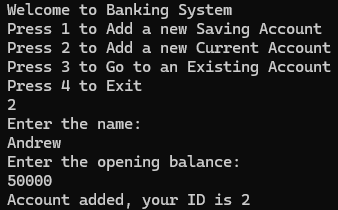
}

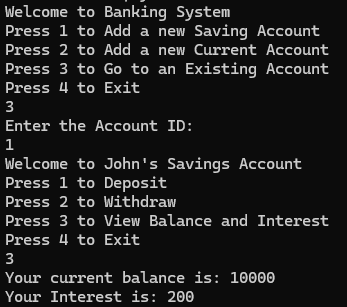
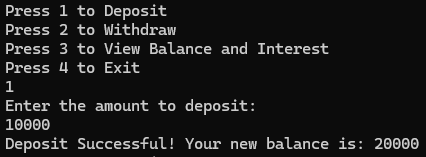
}

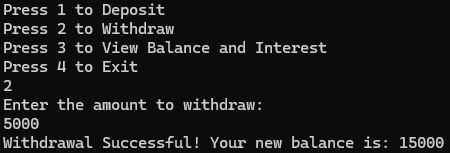
}

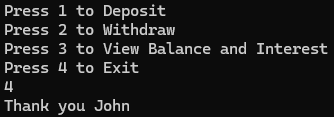
}

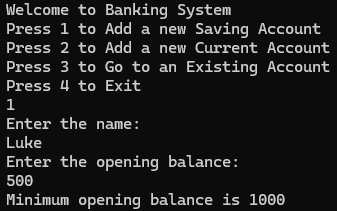
**Output-**







**Program-13 Write a Java program to create an interface Searchable with a method search(String keyword) that searches for a given keyword in a text document. Create two classes Document and WebPage that implement the Searchable interface and provide their own implementations of the search() method.**

using System;

namespace SearchableSystem{

// Abstract class

public abstract class Searchable{

public string Data { get; set; } = "";

public abstract void Search(string keyword);

}

// Document class implementing Searchable

public class Document : Searchable{

public Document(string data){

Data = data;

}

public override void Search(string keyword){

int index = Data.IndexOf(keyword, StringComparison.OrdinalIgnoreCase);

Console.WriteLine($"Index: {index}");

}

}

// WebPage class implementing Searchable

public class WebPage : Searchable{

public WebPage(string data){

Data = data;

}

public override void Search(string keyword){

int index = Data.IndexOf(keyword, StringComparison.OrdinalIgnoreCase);

if (index > -1){

Console.WriteLine($"Index: {index}");

}

else{

Console.WriteLine("Not found");

}

}

}

// Main Program

class Program{

static void Main(string[] args){

Console.WriteLine("Text in Document: Hello how are you my friend");

Console.WriteLine("String to Search: are you");

Document myDoc = new Document("Hello how are you my friend");

myDoc.Search("are you");

Console.WriteLine();

Console.WriteLine("Text in WebPage: <h1>Today the weather is rainy.</h1>");

Console.WriteLine("String to Search: sunny");

WebPage myWebPage = new WebPage("<h1>Today the weather is rainy.</h1>");

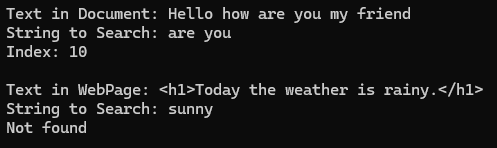
myWebPage.Search("sunny");

}

}

}

**Output-**

****

**Program-14 Write a Java program to create an interface Encryptable with methods encrypt (String data) and decrypt (String encryptedData) that define encryption and decryption operations. Create two classes AES and RSA that implement the Encryptable interface and provide their own encryption and decryption algorithms.**

using System;

namespace EncryptionSystem{

// Interface

public interface IEncryptable{

string Encrypt(string data);

string Decrypt(string data);

}

// AES Algorithm class

public class AES : IEncryptable{

private const int ShiftSteps = 3;

public string Encrypt(string data){

string encryptedData = "";

for (int i = 0; i < data.Length; i++){

encryptedData += (char)(data[i] + ShiftSteps);

}

return encryptedData;

}

public string Decrypt(string data){

string decryptedData = "";

for (int i = 0; i < data.Length; i++){

decryptedData += (char)(data[i] - ShiftSteps);

}

return decryptedData;

}

}

// RSA Algorithm class

public class RSA : IEncryptable{

private const int ShiftSteps = 1;

public string Encrypt(string data){

string encryptedData = "";

for (int i = 0; i < data.Length; i++){

encryptedData += (char)(data[i] + ShiftSteps);

}

return encryptedData;

}

public string Decrypt(string data){

string decryptedData = "";

for (int i = 0; i < data.Length; i++){

decryptedData += (char)(data[i] - ShiftSteps);

}

return decryptedData;

}

}

// Main Program

class Program{

static void Main(string[] args){

AES algo1 = new AES();

RSA algo2 = new RSA();

string data = "The London Bridge is burning.";

Console.WriteLine($"Text to Cipher: {data}");

Console.WriteLine();

string aesEnc = algo1.Encrypt(data);

string rsaEnc = algo2.Encrypt(data);

Console.WriteLine($"After Encryption");

Console.WriteLine($"AES: {aesEnc}");

Console.WriteLine($"RSA: {rsaEnc}");

Console.WriteLine();

Console.WriteLine($"After Decryption");

Console.WriteLine($"AES: {algo1.Decrypt(aesEnc)}");

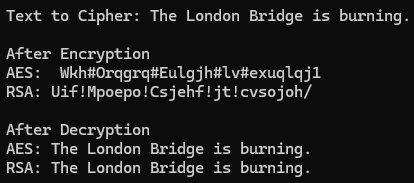
Console.WriteLine($"RSA: {algo2.Decrypt(rsaEnc)}");

}

}

}

**Output-**

****

**Program-15 Write a program to manage the employees of an organization. The employees are categorized as**

**A) Developer:**

* **Salary**
* **Including basic + TA + DA + HTA + PPF**
* **Incentives**

**B) Back Office Worker:**

* **Salary**
* **Including basic + TA + DA + HTA + PPF**

**C) Tester:**

* **Salary**
* **Including basic + TA + DA + HTA + PPF**
* **Incentives on the basis of no. of projects tested**
* **Incentives on the basis of no, of reports given**

**D) Marketeer:**

* **Salary**
* **Including basic + TA + DA + HTA + PPF**
* **Incentives**

**Salary should be calculated on behalf of attendance. All the types of employees should extend from an abstract class.**

//-------------------EMPLOYEE (abstract)----------------->

public interface Employee{

string Id { get; set; }

string Name { get; set; }

float Package { get; set; } //input in LPA

int Attendance { get; set; }

int Da { get; set; }

int Ta { get; set; }

int Hra { get; set; }

int Ppf { get; set; }

float GrossSalary { get; set; }

float NetSalary { get; set; }

float TotalSalary { get; set; }

//acc to attendance

float CalcGrossSalary();

//acc to gross + ta + da + hra - ppf

float CalcNetSalary();

//netsal + incentives

float CalcTotalSalary();

void PrintSlip();

}

//-------------------DEVELOPER----------------->

public class Dev : Employee{

public string Id { get; set; }

public string Name { get; set; }

public float Package { get; set; }

public int Attendance { get; set; }

public int Da { get; set; } = 2000;

public int Ta { get; set; } = 2000;

public int Hra { get; set; } = 15000;

public int Ppf { get; set; } = 550;

public int Incentives { get; set; } = 10000;

public float GrossSalary { get; set; }

public float NetSalary { get; set; }

public float TotalSalary { get; set; }

public float CalcGrossSalary(){

return Package \* (Attendance / 31f);

}

public float CalcNetSalary(){

return GrossSalary + Da + Ta + Hra - Ppf;

}

public float CalcTotalSalary(){

return NetSalary + Incentives;

}

public void PrintSlip(){

Console.WriteLine();

Console.WriteLine("---------------Pay Slip--------------");

Console.WriteLine($"Salary Slip for {Name} ({Id}):");

Console.WriteLine("Role: Developer");

Console.WriteLine($"Base Package: {Package}/mo");

Console.WriteLine($"Attendance out of 31: {Attendance}");

Console.WriteLine($"DA: {Da}");

Console.WriteLine($"TA: {Ta}");

Console.WriteLine($"HRA: {Hra}");

Console.WriteLine($"PPF: {Ppf}");

Console.WriteLine($"Gross Salary (acc to attendance): {GrossSalary}");

Console.WriteLine($"Net Salary (GrossSal + Da + Ta + Hra - Ppf: {NetSalary}");

Console.WriteLine($"Developer Specific Incentives: {Incentives}");

Console.WriteLine($"Total Salary (Net Salary + Incentives): {TotalSalary}");

Console.WriteLine();

}

//Constructor

public Dev(string id, string name, float package, int attendance){

Id = id;

Name = name;

Package = (package\*100000)/12;

Attendance = attendance;

GrossSalary = CalcGrossSalary();

NetSalary = CalcNetSalary();

TotalSalary = CalcTotalSalary();

}

}

//-------------------BACKOFFICE----------------->

public class BackOffice : Employee{

public string Id { get; set; }

public string Name { get; set; }

public float Package { get; set; }

public int Attendance { get; set; }

public int Da { get; set; } = 1000;

public int Ta { get; set; } = 1000;

public int Hra { get; set; } = 8000;

public int Ppf { get; set; } = 1050;

public int Incentives { get; set; } = 0; // 0 bcoz no Incentives to BackOffice workers

public float GrossSalary { get; set; }

public float NetSalary { get; set; }

public float TotalSalary { get; set; }

public float CalcGrossSalary(){

return Package \* (Attendance / 31f);

}

public float CalcNetSalary(){

return GrossSalary + Da + Ta + Hra - Ppf;

}

public float CalcTotalSalary(){

return NetSalary + Incentives;

}

public void PrintSlip(){

Console.WriteLine();

Console.WriteLine("---------------Pay Slip--------------");

Console.WriteLine($"Salary Slip for {Name} ({Id}):");

Console.WriteLine("Role: BackOffice");

Console.WriteLine($"Base Package: {Package}/mo");

Console.WriteLine($"Attendance out of 31: {Attendance}");

Console.WriteLine($"DA: {Da}");

Console.WriteLine($"TA: {Ta}");

Console.WriteLine($"HRA: {Hra}");

Console.WriteLine($"PPF: {Ppf}");

Console.WriteLine($"Gross Salary (acc to attendance): {GrossSalary}");

Console.WriteLine($"Net Salary (GrossSal + Da + Ta + Hra - Ppf: {NetSalary}");

Console.WriteLine($"BackOffice Specific Incentives: {Incentives}");

Console.WriteLine($"Total Salary (Net Salary + Incentives): {TotalSalary}");

Console.WriteLine();

}

//Constructor

public BackOffice(string id, string name, float package, int attendance){

Id = id;

Name = name;

Package = (package \* 100000) / 12;

Attendance = attendance;

GrossSalary = CalcGrossSalary();

NetSalary = CalcNetSalary();

TotalSalary = CalcTotalSalary();

}

}

//-------------------TESTER----------------->

public class Tester : Employee{

public string Id { get; set; }

public string Name { get; set; }

public float Package { get; set; }

public int Attendance { get; set; }

public int Da { get; set; } = 1500;

public int Ta { get; set; } = 3000;

public int Hra { get; set; } = 6000;

public int Ppf { get; set; } = 3000;

public int Incentives { get; set; }

public int noProjTested { get; set; } //Tester specific property

public int noReports { get; set; } //Tester specific property

public float GrossSalary { get; set; }

public float NetSalary { get; set; }

public float TotalSalary { get; set; }

int CalcIncentives() //Tester specific method{

if (this.noReports > 10 || this.noProjTested > 10) return 10000;

else if (this.noReports > 5 || this.noProjTested > 5) return 5000;

else return 1000;

}

public float CalcGrossSalary(){

return Package \* (Attendance / 31f);

}

public float CalcNetSalary(){

return GrossSalary + Da + Ta + Hra - Ppf;

}

public float CalcTotalSalary(){

return NetSalary + Incentives;

}

public void PrintSlip(){

Console.WriteLine();

Console.WriteLine("---------------Pay Slip--------------");

Console.WriteLine($"Salary Slip for {Name} ({Id}):");

Console.WriteLine("Role: Tester");

Console.WriteLine($"Base Package: {Package}/mo");

Console.WriteLine($"Attendance out of 31: {Attendance}");

Console.WriteLine($"DA: {Da}");

Console.WriteLine($"TA: {Ta}");

Console.WriteLine($"HRA: {Hra}");

Console.WriteLine($"PPF: {Ppf}");

Console.WriteLine($"Gross Salary (acc to attendance): {GrossSalary}");

Console.WriteLine($"Net Salary (GrossSal + Da + Ta + Hra - Ppf: {NetSalary}");

Console.WriteLine($"Tester Specific Incentives (acc to proj tested and reports given): {Incentives}");

Console.WriteLine($"Total Salary (Net Salary + Incentives): {TotalSalary}");

Console.WriteLine();

}

//Constructor

public Tester(string id, string name, float package, int attendance, int noProjTested, int noReports){

Id = id;

Name = name;

Package = (package \* 100000) / 12;

Attendance = attendance;

this.noProjTested = noProjTested;

this.noReports = noReports;

Incentives = CalcIncentives();

GrossSalary = CalcGrossSalary();

NetSalary = CalcNetSalary();

TotalSalary = CalcTotalSalary();

}

}

//-------------------MKT----------------->

public class Mkt : Employee{

public string Id { get; set; }

public string Name { get; set; }

public float Package { get; set; }

public int Attendance { get; set; }

public int Da { get; set; } = 500;

public int Ta { get; set; } = 30;

public int Hra { get; set; } = 6;

public int Ppf { get; set; } = 6000;

public int Incentives { get; set; } = -150;

public float GrossSalary { get; set; }

public float NetSalary { get; set; }

public float TotalSalary { get; set; }

public float CalcGrossSalary(){

return Package \* (Attendance / 31f);

}

public float CalcNetSalary(){

return GrossSalary + Da + Ta + Hra - Ppf;

}

public float CalcTotalSalary(){

return NetSalary + Incentives;

}

public void PrintSlip(){

Console.WriteLine();

Console.WriteLine("---------------Pay Slip--------------");

Console.WriteLine($"Salary Slip for {Name} ({Id}):");

Console.WriteLine("Role: Mkt");

Console.WriteLine($"Base Package: {Package}/mo");

Console.WriteLine($"Attendance out of 31: {Attendance}");

Console.WriteLine($"DA: {Da}");

Console.WriteLine($"TA: {Ta}");

Console.WriteLine($"HRA: {Hra}");

Console.WriteLine($"PPF: {Ppf}");

Console.WriteLine($"Gross Salary (acc to attendance): {GrossSalary}");

Console.WriteLine($"Net Salary (GrossSal + Da + Ta + Hra - Ppf: {NetSalary}");

Console.WriteLine($"Mkt Specific Incentives: {Incentives}");

Console.WriteLine($"Total Salary (Net Salary + Incentives): {TotalSalary}");

Console.WriteLine();

}

//Constructor

public Mkt(string id, string name, float package, int attendance){

Id = id;

Name = name;

Package = (package \* 100000) / 12;

Attendance = attendance;

GrossSalary = CalcGrossSalary();

NetSalary = CalcNetSalary();

TotalSalary = CalcTotalSalary();

}

}

class Execution {

//Data structures

public static Dev[] Devs;

public static BackOffice[] BackOffices;

public static Tester[] Testers;

public static Mkt[] Mkts;

public static int globalUniqueId = 1;

//Making UI

public static void Main(string[] args){

//Taking All Employees' Details

Console.WriteLine("Enter the team details of org");

//----------DEVELOPER--------->

Console.WriteLine("How many Devs?: ");

int noDevs = int.Parse(Console.ReadLine()!);

Devs = new Dev[noDevs];

for (int i = 0; i < noDevs; i++){

string id = globalUniqueId++.ToString();

Console.Write("Enter Dev's Name: ");

string name = Console.ReadLine()!;

Console.Write("Enter Package in LPA like 12, 8: ");

float pkg = float.Parse(Console.ReadLine()!);

Console.Write("Enter attendance 0-31: ");

int attendance = int.Parse(Console.ReadLine()!);

Devs[i] = new Dev(id, name, pkg, attendance);

}

//----------BACKOFFICE--------->

Console.WriteLine("How many BackOffice workers?: ");

int noBackOffice = int.Parse(Console.ReadLine()!);

BackOffices = new BackOffice[noBackOffice];

for (int i = 0; i < noBackOffice; i++){

string id = globalUniqueId++.ToString();

Console.Write("Enter BackOffice workers's Name: ");

string name = Console.ReadLine()!;

Console.Write("Enter Package in LPA like 12, 8: ");

float pkg = float.Parse(Console.ReadLine()!);

Console.Write("Enter attendance 0-31: ");

int attendance = int.Parse(Console.ReadLine()!);

BackOffices[i] = new BackOffice(id, name, pkg, attendance);

}

//----------TESTER--------->

Console.WriteLine("How many Testers?: ");

int noTesters = int.Parse(Console.ReadLine()!);

Testers = new Tester[noTesters];

for (int i = 0; i < noTesters; i++){

string id = globalUniqueId++.ToString();

Console.Write("Enter Tester's Name: ");

string name = Console.ReadLine()!;

Console.Write("Enter Package in LPA like 12, 8: ");

float pkg = float.Parse(Console.ReadLine()!);

Console.Write("Enter attendance 0-31: ");

int attendance = int.Parse(Console.ReadLine()!);

Console.Write("Enter No of Projects Tested like 12: ");

int noProjTested = int.Parse(Console.ReadLine()!);

Console.Write("Enter No of Reports Given like 7: ");

int noReports = int.Parse(Console.ReadLine()!);

Testers[i] = new Tester(id, name, pkg, attendance, noProjTested, noReports);

}

//----------MKT--------->

Console.WriteLine("How many Marketing workers?: ");

int noMkts = int.Parse(Console.ReadLine()!);

Mkts = new Mkt[noMkts];

for (int i = 0; i < noMkts; i++){

string id = globalUniqueId++.ToString();

Console.Write("Enter Worker's Name: ");

string name = Console.ReadLine()!;

Console.Write("Enter Package in LPA like 4: ");

float pkg = float.Parse(Console.ReadLine()!);

Console.Write("Enter attendance 0-31: ");

int attendance = int.Parse(Console.ReadLine()!);

Mkts[i] = new Mkt(id, name, pkg, attendance);

}

if (globalUniqueId == 1){

Console.WriteLine("No entried Made");

return;

//exit coz no entries made

}

//Drawing Menu to view Employees' PaySlip

while (true){

Console.Write("Enter the Worker's Id: ");

int id = int.Parse(Console.ReadLine()!);

if (id < 1 || id >= globalUniqueId){

Console.WriteLine("Give valid ID");

continue;

//reprompt for id

}

//find the employee

bool found = false;

if (!found){

for (int i = 0; i < Devs.Length; i++){

if (Devs[i].Id == id.ToString()){

found = true;

Devs[i].PrintSlip();

}

}

}

if (!found){

for (int i = 0; i < BackOffices.Length; i++){

if (BackOffices[i].Id == id.ToString()){

found = true;

BackOffices[i].PrintSlip();

}

}

}

if (!found){

for (int i = 0; i < Testers.Length; i++){

if (Testers[i].Id == id.ToString()){

found = true;

Testers[i].PrintSlip();

}

}

}

if (!found){

for (int i = 0; i < Mkts.Length; i++){

if (Mkts[i].Id == id.ToString()){

found = true;

Mkts[i].PrintSlip();

}

}

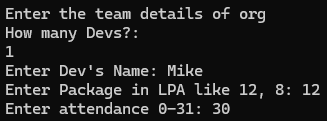
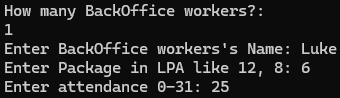
}

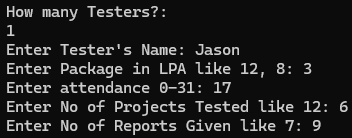
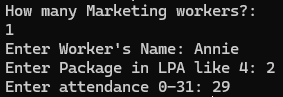
}

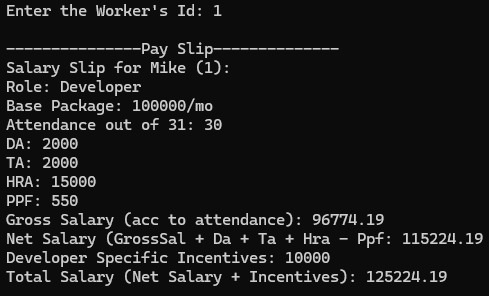
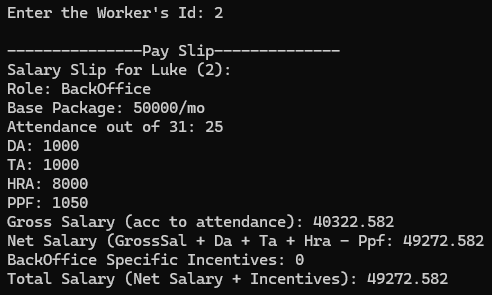
}

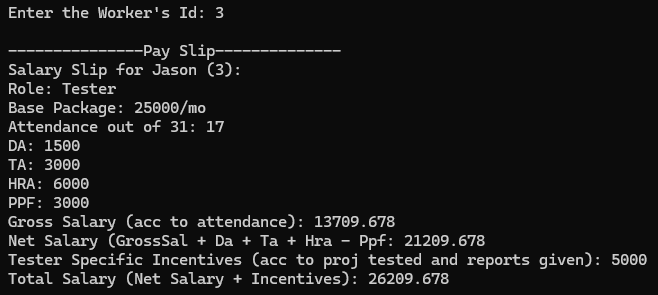
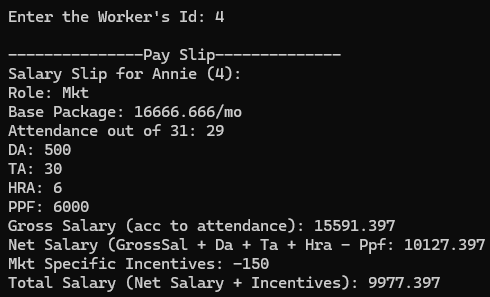
}

**Output-**

**Program-16 Write a program to manage users using windows forms. The program should be able to add users with their username and password. After adding, users can also login. Authenticate their credentials based on the data stored during user creation.**

namespace WinFormsApp2{

public partial class Form1 : Form{

Database db;

int id;

public Form1(){

InitializeComponent();

db = new Database();

id = 0;

lblAddUser.Visible = false;

lblError.Visible = false;

}

//Add User

private void button1\_Click(object sender, EventArgs e){

string username = textBox1.Text;

string password = textBox2.Text;

bool res = db.addUser(new User(id.ToString(), username, password));

lblAddUser.Visible = true;

if (res){

lblAddUser.ForeColor = Color.Green;

lblAddUser.Text = "New User Added";

}

else{

lblAddUser.ForeColor = Color.DarkRed;

lblAddUser.Text = "No more space";

}

}

//Login

private void btnLogin\_Click(object sender, EventArgs e){

string username = txtUsername.Text;

string password = txtUserpassword.Text;

lblError.Visible = true;

if (db.isValid(username, password)){

lblError.ForeColor = Color.Green;

lblError.Text = "Successfully Logged In";

}

else{

lblError.ForeColor = Color.DarkRed;

lblError.Text = "Invalid Credentials";

}

}

private void button1\_Click\_1(object sender, EventArgs e){

Form2 form2 = new Form2();

form2.ShowDialog();

this.Close(); //UI remains, but it is closed in memory

}

}

//User class

public class User{

public string id;

public string name;

public string password;

public User(string id, string name, string password){

this.id = id;

this.name = name;

this.password = password;

}

}

public class Database{

User[] data;

int ptr;

public Database(){

data = new User[2];

ptr = 0;

}

public bool addUser(User user){

if (ptr == data.Length) return false;

data[ptr++] = user;

return true;

}

public bool isValid(String name, String password){

for (int i = 0; i < ptr; i++){

if (data[i].name == name && data[i].password == password){

return true;

}

}

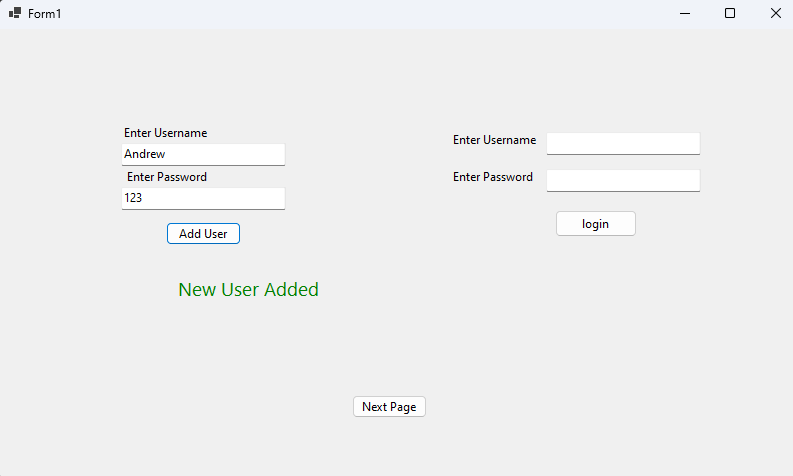
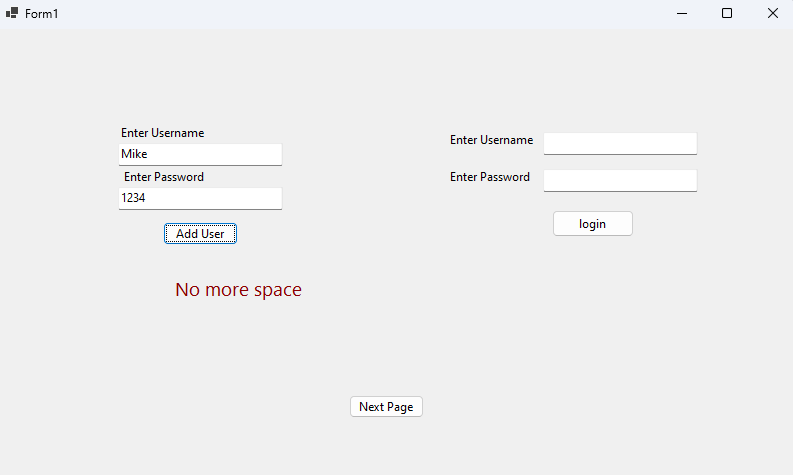
return false;

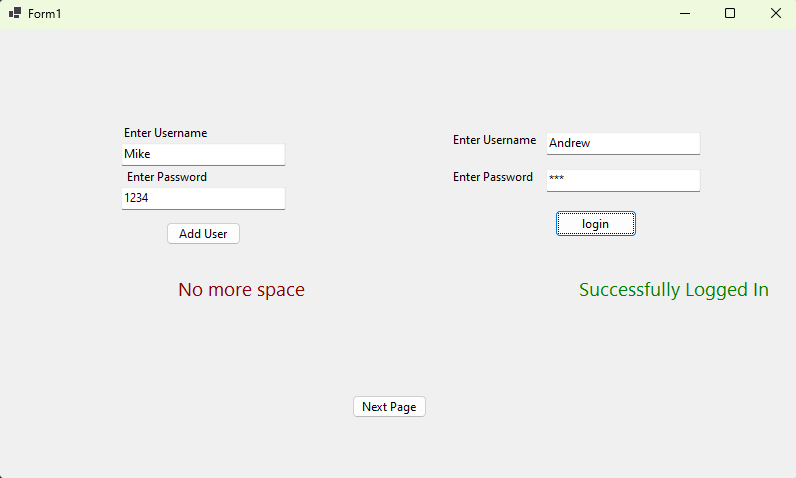
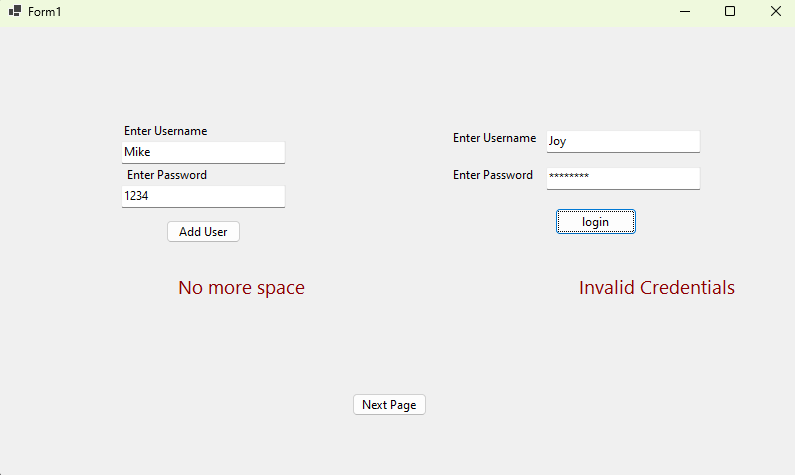
}

}

}

**Ouput-**

** **

** **