**Exercise 1: Inventory Management System**

import java.util.\*;

class Product {

int productId;

String productName;

int quantity;

double price;

Product(int productId, String productName, int quantity, double price) {

this.productId = productId;

this.productName = productName;

this.quantity = quantity;

this.price = price;

}

}

class InventoryManagementSystem {

ArrayList<Product> al = new ArrayList<>();

void addProd(Product p) {

al.add(p);

}

void updateProd(int id, int quantity) {

for (Product p : al) {

if (p.productId == id) {

p.quantity = quantity;

return;

}

}

}

void delProd(int id) {

al.removeIf(p -> p.productId == id);

}

void display()

{

System.out.println("Inventory:");

for (Product p : al) {

System.out.println("ID: " + p.productId + ", Name: " + p.productName + ", Qty: " + p.quantity + ", Price: " + p.price);

}

}

}

public class Exercise1

{

public static void main(String[] args)

{

InventoryManagementSystem obj = new InventoryManagementSystem();

obj.addProd(new Product(1, "Pen", 50, 5.5));

obj.addProd(new Product(2, "Notebook", 30, 15.0));

obj.addProd(new Product(3, "Pencil", 100, 2.0));

obj.display();

obj.updateProd(2, 40);

obj.delProd(1);

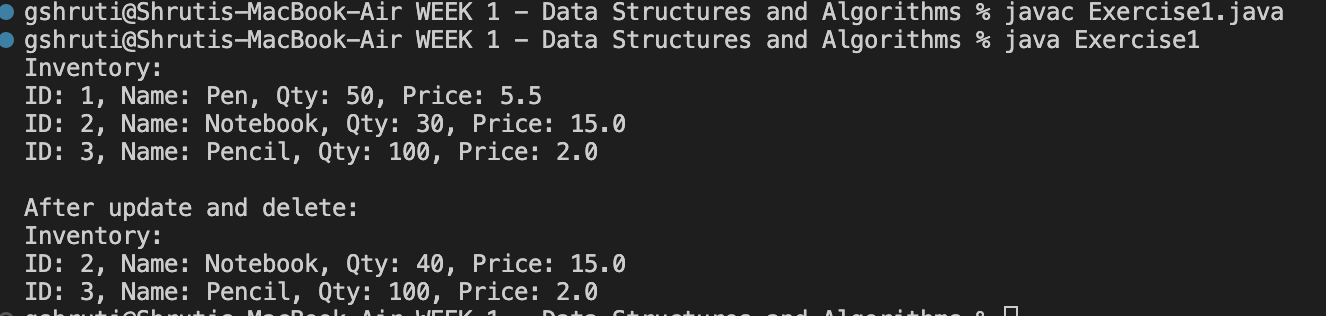
System.out.println("\nAfter update and delete:");

obj.display();

}

}

Output:



**Exercise 2: E-commerce Platform Search Function**

import java.util.Arrays;

import java.util.Scanner;

class Product {

int productId;

String productName;

String category;

Product(int productId, String productName, String category) {

this.productId = productId;

this.productName = productName;

this.category = category;

}

}

public class Exercise2

{

public static int linearSearch(Product[] p, String k)

{

for (int i = 0; i < p.length; i++) {

if (p[i].productName.equalsIgnoreCase(k)) {

return i;

}

}

return -1;

}

public static int binarySearch(Product[] p, String k)

{

int low = 0, high = p.length - 1;

while (low <= high) {

int mid = (low + high) / 2;

int res = p[mid].productName.compareToIgnoreCase(k);

if (res == 0) return mid;

else if (res < 0) low = mid + 1;

else high = mid - 1;

}

return -1;

}

public static void main(String[] args)

{

Product[] products =

{

new Product(101, "Laptop", "Electronics"),

new Product(102, "Shoes", "Footwear"),

new Product(103, "Phone", "Electronics"),

new Product(104, "Watch", "Accessories")

};

Arrays.sort(products, new java.util.Comparator<Product>()

{

public int compare(Product a, Product b)

{

return a.productName.compareToIgnoreCase(b.productName);

}

});

Scanner sc = new Scanner(System.in);

System.out.print("Enter product name to search: ");

String key = sc.nextLine();

int lind = linearSearch(products, key);

int bind = binarySearch(products, key);

if (lind != -1)

{

System.out.println("Linear Search: Found at index " + lind);

}

else

{

System.out.println("Linear Search: Product not found");

}

if (bind != -1) {

System.out.println("Binary Search: Found at index " + bind);

} else {

System.out.println("Binary Search: Product not found");

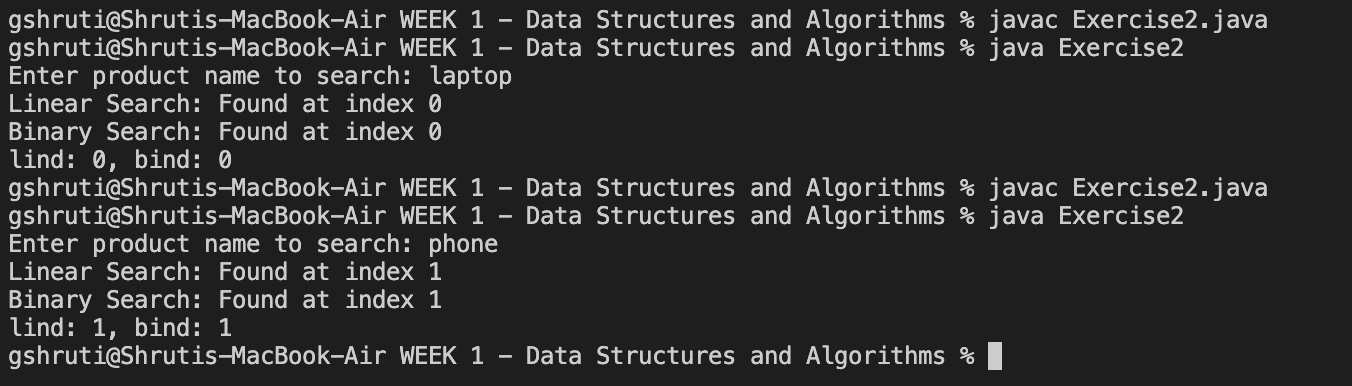
}

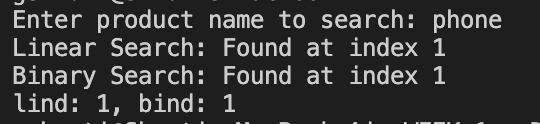
System.out.println("lind: " + lind + ", bind: " + bind);

sc.close();

}

}

Output   
****

****

**Exercise 3: Sorting Customer Orders**

class Order {

int orderId;

String customerName;

double totalPrice;

Order(int orderId, String customerName, double totalPrice) {

this.orderId = orderId;

this.customerName = customerName;

this.totalPrice = totalPrice;

}

void print() {

System.out.println("OrderID: " + orderId + ", Name: " + customerName + ", Total: " + totalPrice);

}

}

class OrderSorting

{

void bubbleSort(Order[] arr){

int n = arr.length;

for (int i = 0; i < n - 1; i++)

{

for (int j = 0; j < n - i - 1; j++)

{

if (arr[j].totalPrice > arr[j + 1].totalPrice)

{

Order t = arr[j];

arr[j] = arr[j + 1];

arr[j + 1] = t;

}

}

}

}

void quickSort(Order[] arr, int low, int high)

{

if (low < high)

{

int pi = partition(arr, low, high);

quickSort(arr, low, pi - 1);

quickSort(arr, pi + 1, high);

}

}

int partition(Order[] arr, int low, int high) {

double pivot = arr[high].totalPrice;

int i = low - 1;

for (int j = low; j < high; j++) {

if (arr[j].totalPrice < pivot) {

i++;

Order temp = arr[i];

arr[i] = arr[j];

arr[j] = temp;

}

}

Order temp = arr[i + 1];

arr[i + 1] = arr[high];

arr[high] = temp;

return i + 1;

}

}

public class Exercise3 {

public static void main(String[] args) {

Order[] orders = {

new Order(1, "Alice", 1200.50),

new Order(2, "Bob", 950.00),

new Order(3, "Charlie", 1500.75),

new Order(4, "David", 500.20)

};

System.out.println("Original Orders:");

for (Order o : orders) o.print();

OrderSorting s = new OrderSorting();

Order[] bs = orders.clone();

s.bubbleSort(bs);

System.out.println("\nOrders sorted by Bubble Sort:");

for (Order o : bs) o.print();

Order[] qs = orders.clone();

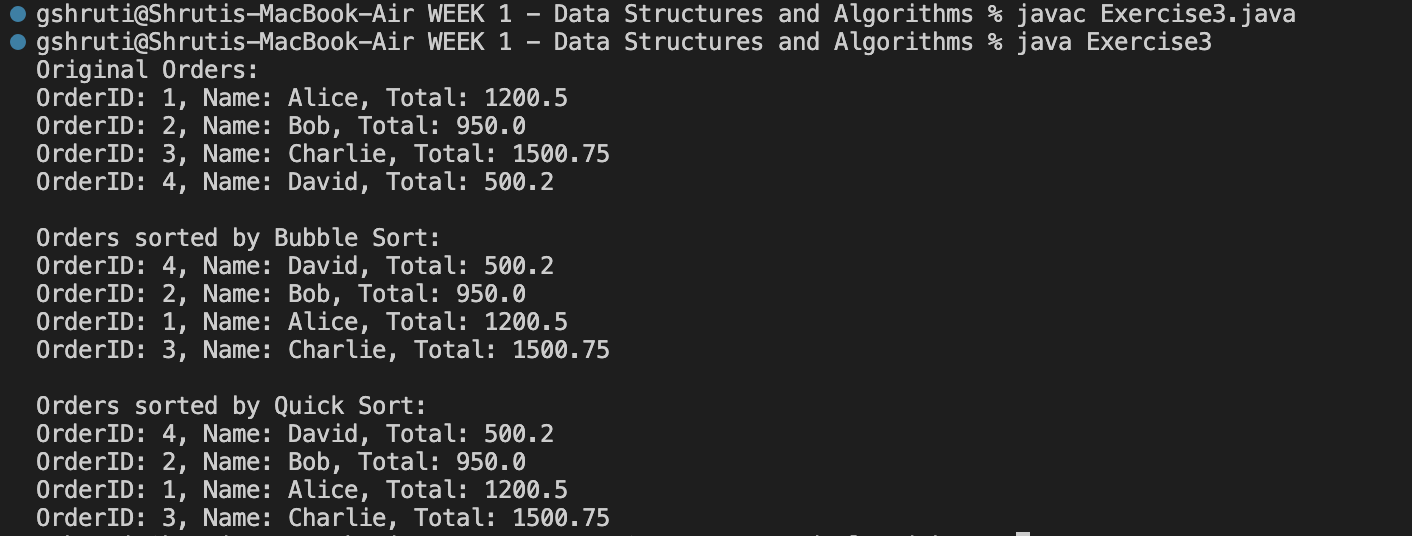
s.quickSort(qs, 0, qs.length - 1);

System.out.println("\nOrders sorted by Quick Sort:");

for (Order o : qs) o.print();

}

}

Output:   


**Exercise 4: Employee Management System**

class Employee {

int employeeId;

String name;

String position;

double salary;

Employee(int employeeId, String name, String position, double salary) {

this.employeeId = employeeId;

this.name = name;

this.position = position;

this.salary = salary;

}

void print() {

System.out.println(employeeId + ": " + name + " - " + position + " ($" + salary + ")");

}

}

class EmployeeSystem {

Employee[] arr = new Employee[100];

int count = 0;

void add(Employee e) {

if (count < arr.length) {

arr[count++] = e;

}

}

void search(int id) {

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

if (arr[i].employeeId == id) {

System.out.println("Employee Found:");

arr[i].print();

return;

}

}

System.out.println("Employee not found.");

}

void del(int id) {

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

if (arr[i].employeeId == id) {

for (int j = i; j < count - 1; j++) {

arr[j] = arr[j + 1];

}

count--;

System.out.println("Employee deleted.");

return;

}

}

System.out.println("Employee not found.");

}

void display() {

if (count == 0) {

System.out.println("No employees in the system.");

return;

}

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

arr[i].print();

}

}

}

public class Exercise4 {

public static void main(String[] args) {

EmployeeSystem es = new EmployeeSystem();

es.add(new Employee(101, "Alice", "Manager", 75000));

es.add(new Employee(102, "Bob", "Developer", 60000));

es.add(new Employee(103, "Charlie", "Analyst", 55000));

System.out.println("All Employees:");

es.display();

System.out.println("\nSearching for Employee ID 102:");

es.search(102);

System.out.println("\nDeleting Employee ID 101:");

es.del(101);

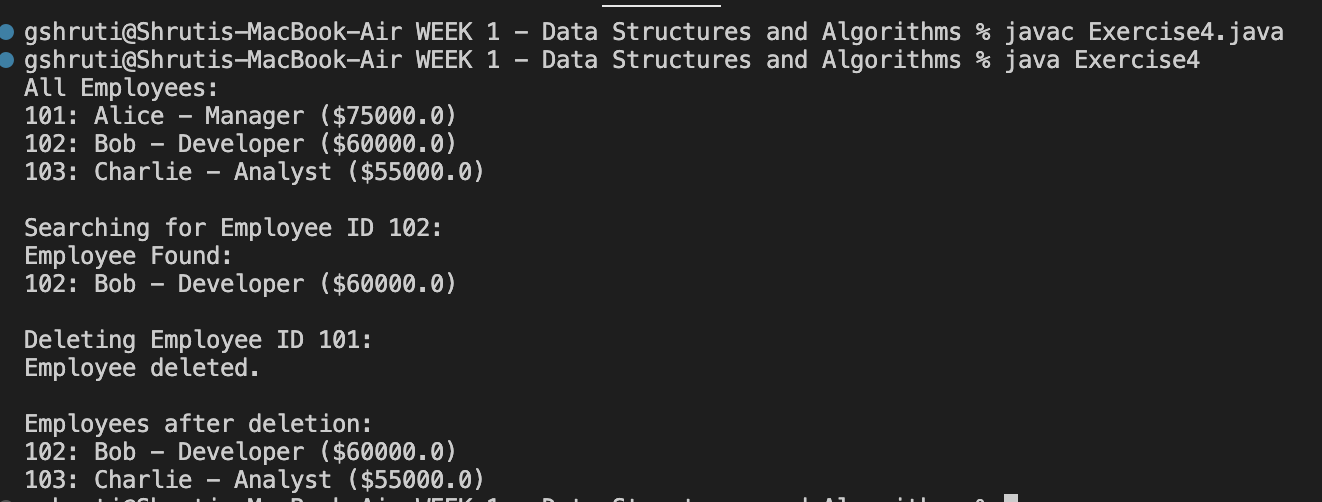
System.out.println("\nEmployees after deletion:");

es.display();

}

}

Output:



**Exercise 5: Task Management System**

class Task {

int taskId;

String taskName;

String status;

Task next;

Task(int taskId, String taskName, String status) {

this.taskId = taskId;

this.taskName = taskName;

this.status = status;

this.next = null;

}

void print() {

System.out.println(taskId + ": " + taskName + " [" + status + "]");

}

}

class TaskList {

Task head = null;

void add(Task task) {

if (head == null) {

head = task;

} else {

Task temp = head;

while (temp.next != null) {

temp = temp.next;

}

temp.next = task;

}

}

void del(int id) {

if (head == null) return;

if (head.taskId == id) {

head = head.next;

return;

}

Task prev = null, curr = head;

while (curr != null && curr.taskId != id) {

prev = curr;

curr = curr.next;

}

if (curr != null) {

prev.next = curr.next;

}

}

void search(int id) {

Task temp = head;

while (temp != null) {

if (temp.taskId == id) {

System.out.println("Found Task:");

temp.print();

return;

}

temp = temp.next;

}

System.out.println("Task not found.");

}

void display() {

Task temp = head;

while (temp != null) {

temp.print();

temp = temp.next;

}

}

}

public class Exercise5 {

public static void main(String[] args) {

TaskList list = new TaskList();

list.add(new Task(1, "Design UI", "Pending"));

list.add(new Task(2, "Build Backend", "In Progress"));

list.add(new Task(3, "Write Tests", "Pending"));

System.out.println("All Tasks:");

list.display();

System.out.println("\nSearching for Task ID 2:");

list.search(2);

System.out.println("\nDeleting Task ID 1:");

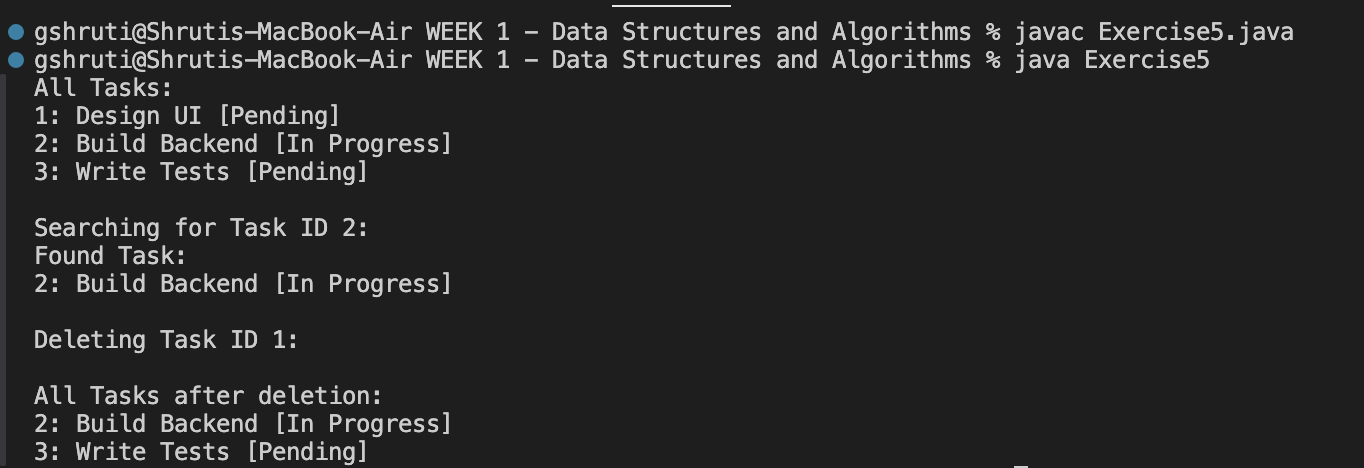
list.del(1);

System.out.println("\nAll Tasks after deletion:");

list.display();

}

}

Output: 

**Exercise 6: Library Management System**

import java.util.Arrays;

class Book {

int bookId;

String title;

String author;

Book(int bookId, String title, String author) {

this.bookId = bookId;

this.title = title;

this.author = author;

}

void print() {

System.out.println(bookId + ": " + title + " by " + author);

}

}

class LibrarySystem {

Book[] b;

LibrarySystem(Book[] b) {

this.b = b;

}

int linearSearch(String s) {

for (int i = 0; i < b.length; i++) {

if (b[i].title.equalsIgnoreCase(s)) {

return i;

}

}

return -1;

}

int binarySearch(String s) {

int low = 0, high = b.length - 1;

while (low <= high) {

int mid = (low + high) / 2;

int cmp = b[mid].title.compareToIgnoreCase(s);

if (cmp == 0) return mid;

else if (cmp < 0) low = mid + 1;

else high = mid - 1;

}

return -1;

}

}

public class Exercise6 {

public static void main(String[] args) {

Book[] arr = {

new Book(1, "Algorithms", "CLRS"),

new Book(2, "Data Structures", "Narasimha Karumanchi"),

new Book(3, "Java Programming", "Herbert Schildt"),

new Book(4, "Operating Systems", "Galvin")

};

Arrays.sort(arr, (b1, b2) -> b1.title.compareToIgnoreCase(b2.title));

LibrarySystem lib = new LibrarySystem(arr);

System.out.println("All Books:");

for (Book b : arr) b.print();

String s = "Java Programming";

int lind = lib.linearSearch(s);

System.out.println("\nLinear Search Result:");

System.out.println(lind >= 0 ? "Found: " + arr[lind].title : "Not found");

int bind = lib.binarySearch(s);

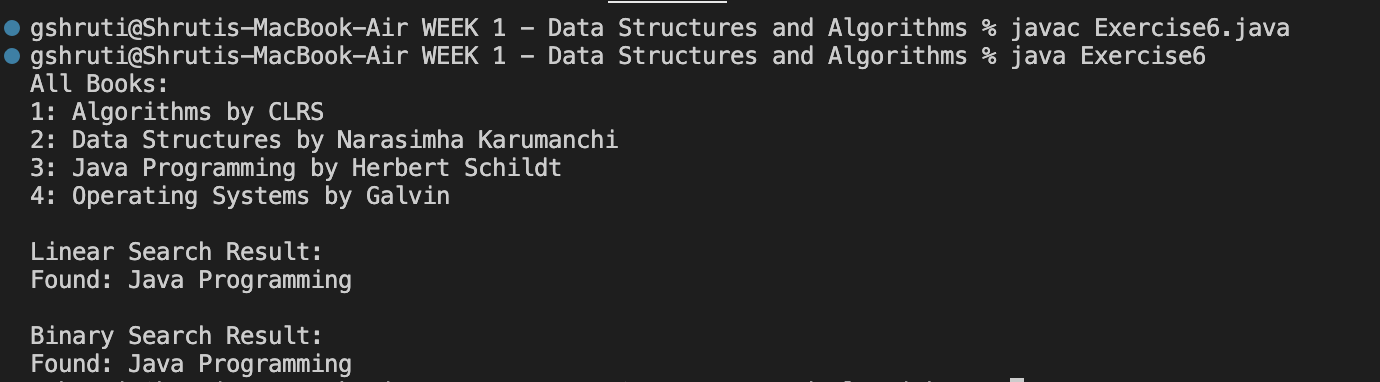
System.out.println("\nBinary Search Result:");

System.out.println(bind >= 0 ? "Found: " + arr[bind].title : "Not found");

}

}

**Output:**



**Exercise 7: Financial Forecasting**

import java.util.Scanner;

public class Exercise7

{

public static double forecastValue(double pv, double r, int y) {

if (y == 0) return pv;

return forecastValue(pv \* (1 + r), r, y - 1);

}

public static double forecastIterative(double pv, double r, int y) {

for (int i = 0; i < y; i++)

{

pv \*= (1 + r);

}

return pv;

}

public static void main(String[] args) {

Scanner sc = new Scanner(System.in);

System.out.print("Enter present value: ");

double pv = sc.nextDouble();

System.out.print("Enter annual growth rate (in %): ");

double r = sc.nextDouble() / 100;

System.out.print("Enter number of years: ");

int y = sc.nextInt();

double futureValueRecursive = forecastValue(pv, r, y);

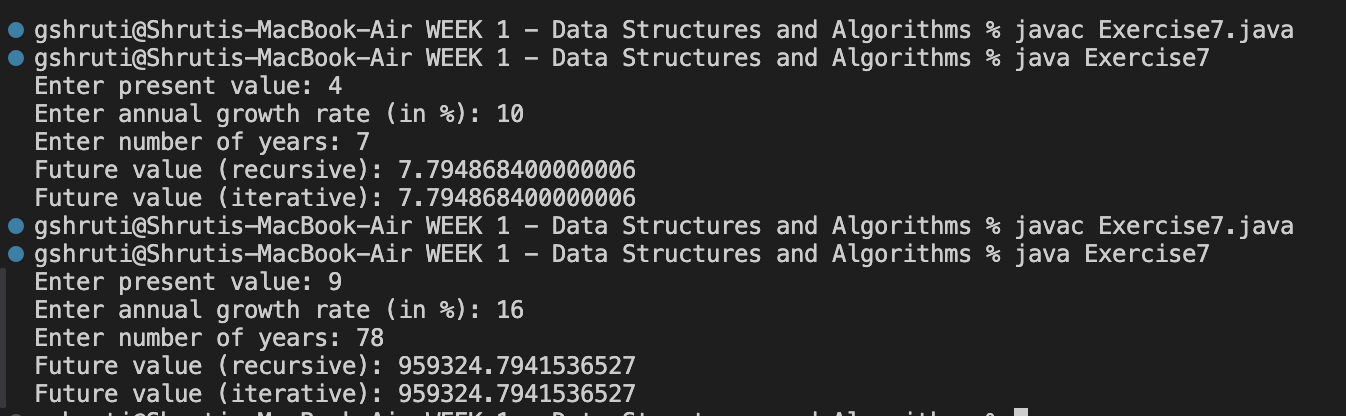
System.out.println("Future value (recursive): " + futureValueRecursive);

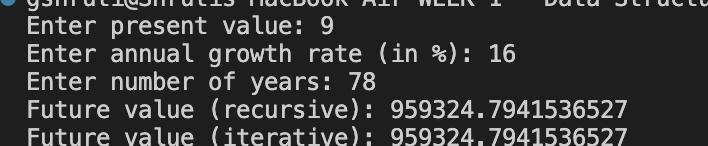
double futureValueIterative = forecastIterative(pv, r, y);

System.out.println("Future value (iterative): " + futureValueIterative);

}

}

Output  
****

****