DSA PRACTICE QUESTIONS - DAY 2

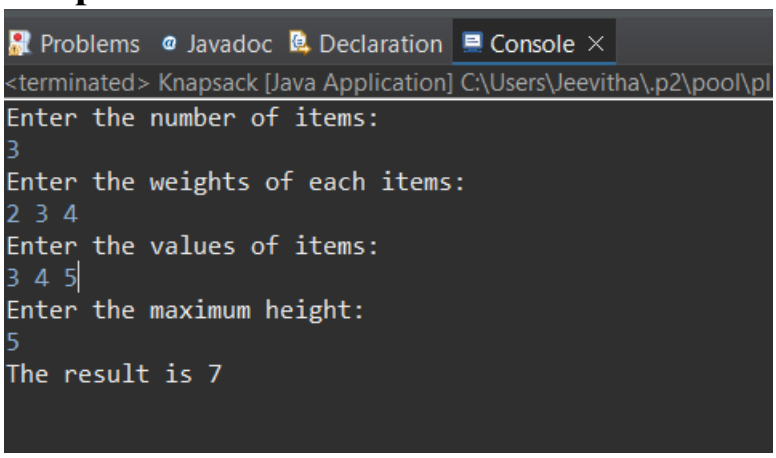
Date: 11/11/2024
Name: Jeevitha R
Reg No: 22IT040

1.Knapsack problem

```
package practiceset2;
import java.util.*;
public class Knapsack {

	public static void main(String[] args) {
		Scanner sc= new Scanner(System.in);
		System.out.println("Enter the number of items:");
		int n= sc.nextInt();
		System.out.println("Enter the weights of each items:");
		int[] weight= new int[n];
		for(int i=0;i<n;i++) {
			weight[i]= sc.nextInt();
		}
		System.out.println("Enter the values of items:");
		int[] value= new int[n];
		for(int i=0;i<n;i++) {
			value[i]= sc.nextInt();
		}
		System.out.println("Enter the maximum height:");
		int Max= sc.nextInt();

		System.out.println("The result is "+ knapsack(weight,value,n,Max));
	}
	public static int knapsack(int[] weight, int[] value, int n, int Max) {
		int[][] dp= new int[n][Max+1];
		for(int[] r: dp) {
			Arrays.fill(r,-1);
		}
		return f(n-1, Max, weight, value, dp);
	}
	public static int f(int index, int Max, int[] weight, int[] value, int[][] dp) {
		if(index==0) {
			if(weight[0]<= Max) {
				return value[0];
			}else {
				return 0;
			}
		}
		if(dp[index][Max]!=-1) {
			return dp[index][Max];
		}
		int notTake= f(index-1, Max, weight, value, dp);
		int take= Integer.MIN_VALUE;
```

```
                if(weight[index]<= Max) {
                        take= value[index]+ f(index-1, Max-weight[index], weight, value, dp);
                }
                return dp[index][Max]= Math.max(take,  notTake);
        }

}
```

Output:



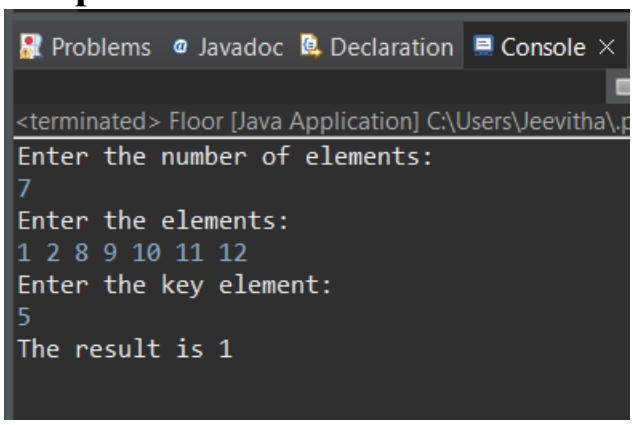
Time complexity: O(N*Max)
Space complexity: O(N*Max)

2.Floor sum

```
package practiceset2;
import java.util.*;
public class Floor {
   public static void main(String[] args){
      Scanner sc= new Scanner(System.in);
      System.out.println("Enter the number of elements:");
      int n= sc.nextInt();
      System.out.println("Enter the elements:");
      int[] arr = new int[n];
      for(int i=0;i<n;i++) {
         arr[i]= sc.nextInt();
      }
      System.out.println("Enter the key element:");
      int k= sc.nextInt();
      int low= 0;
      int high= arr.length - 1;
      int res= -1;
      while(low<= high) {
         int mid= low+(high-low)/2;
         if(arr[mid]== k){
            res= mid;
         }else if(arr[mid]< k) {
            res= mid;
            low= mid+1;
         }else{
            high= mid-1;
```

```
        }
    }
    System.out.println("The result is "+ res);;
  }
  }
```

Output:



Time complexity: O(log n)
Space complexity: O(n)

3.Check arrays are equal

```
package practiceset2;
import java.util.*;
public class EqualArrays {
    public static void main(String[] args) {
    Scanner sc=new Scanner(System.in);
    System.out.print("Enter size of arrays: ");
    int n=sc.nextInt();
    int[] arr1=new int[n];
    int[] arr2=new int[n];
    if(arr1.length!=arr2.length){
      System.out.println("false");
      return;
    }
    HashMap<Integer,Integer> freqMap1=new HashMap<>();
    HashMap<Integer,Integer> freqMap2=new HashMap<>();
    System.out.print("Enter elements of arr1: ");
    for(int i=0;i<n;i++){
      arr1[i]=sc.nextInt();
      freqMap1.put(arr1[i],freqMap1.getOrDefault(arr1[i],0)+1);
    }
    System.out.print("Enter elements of arr2: ");
    for(int i=0;i<n;i++){
      arr2[i]=sc.nextInt();
      freqMap2.put(arr2[i],freqMap2.getOrDefault(arr2[i],0)+1);
    }
    System.out.println(freqMap1.equals(freqMap2));

      }
}
```

Output:

```
Problems  Javadoc  Declaration  Console ×
<terminated> EqualArrays [Java Application] C:\Users\Jeevit
Enter size of arrays: 5
Enter elements of arr1: 2 5 0 4 3
Enter elements of arr2: 3 4 5 2 0
true
```

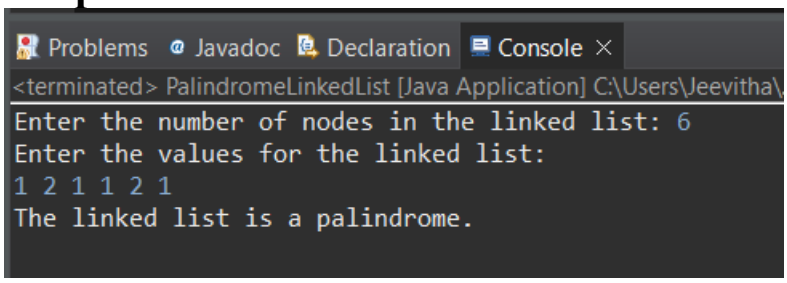
Time complexity: O(n)
Space complexity: O(n)

4.Palindrome Linked List

```
package practiceset2;
import java.util.*;
public class PalindromeLinkedList{
    public static void main(String[] args){
        Scanner sc=new Scanner(System.in);
        System.out.print("Enter the number of nodes in the linked list: ");
        int n=sc.nextInt();
        if(n<=0){
            System.out.println("The list is empty or invalid.");
            return;
        }
        System.out.println("Enter the values for the linked list:");
        Node head=new Node(sc.nextInt());
        Node current=head;
        for(int i=1;i<n;i++){
            int data=sc.nextInt();
            current.next=new Node(data);
            current=current.next;
        }
        Node slow=head,fast=head;
        while(fast!=null&&fast.next!=null){
            slow=slow.next;
            fast=fast.next;
            if(fast!=null) fast=fast.next;
        }
        Node prev=null,curr=slow,nexti=slow.next;
        while(curr!=null){
            curr.next=prev;
            prev=curr;
            curr=nexti;
            if(nexti!=null) nexti=nexti.next;
        }
        Node h2=prev;
        boolean isPalindrome=true;
        Node h1=head;
        while(h1!=null&&h2!=null){
            if(h1.data!=h2.data){
                isPalindrome=false;
                break;
            }
            h1=h1.next;
```

```
            h2=h2.next;
        }
        if(isPalindrome) System.out.println("The linked list is a palindrome.");
        else System.out.println("The linked list is not a palindrome.");
    }
    static class Node{
        int data;
        Node next;
        Node(int data){
            this.data=data;
            this.next=null;
        }
    }
}
```

Output:



Time complexity: O(n)
Space complexity: O(1)