Name – Durwankur Naik

Roll No – I4214

Div – 2

Sub – Information Storage & Retrieval

Practical No – 2

```
import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStreamReader;
import java.util.ArrayList;
public class singlepass {
 public static void main(String[] args) throws IOException{
 BufferedReader stdInpt = new BufferedReader(new
InputStreamReader(System.in));
 System.out.println("Enter the no of Tokens");
 int noOfDocuments=Integer.parseInt(stdInpt.readLine());
 System.out.println("Enter the no of Documents");
 int noOfTokens=Integer.parseInt(stdInpt.readLine());
 System.out.println("Enter the threshhold");
 float threshhold=Float.parseFloat(stdInpt.readLine());
 System.out.println("Enter the Document Token Matrix");
 int [][]input= new int [noOfDocuments][noOfTokens];
 for(int i=0;i {
 for(int j=0;j {
 System.out.println("Enter("+i+","+j+")");
 input[i][j]=Integer.parseInt(stdInpt.readLine());
 }
 }
 SinglePassAlgorithm(noOfDocuments, noOfTokens, threshhold, input);
 }
 private static void SinglePassAlgorithm(int noOfDocuments,int
noOfTokens,float threshhold,int
[][]input)
 {
 int [][] cluster = new int [noOfDocuments][noOfDocuments+1];
 ArrayList clusterRepresentative = new ArrayList();
 cluster [0][0]=1;
 cluster [0][1]=0;
 int noOfClusters=1;
 Float []temp= new Float[noOfTokens];
 temp=convertintArrToFloatArr(input[0]);
 clusterRepresentative.add(temp);
 for(int i=1;i {
 float max=-1;
 int clusterId=-1;
 for(int j=0;j {
 float
similarity=calculateSimilarity(convertintArrToFloatArr(input[i]),clusterRepres
entative.get(j) );
 if(similarity>threshhold)
 {
 if(similarity>max)
 {
 max=similarity;
 clusterId=j;
 }
 }
 }
 if(max==-1)
 {
 cluster[noOfClusters][0]=1;
 cluster[noOfClusters][1]=i;
 noOfClusters++;
 clusterRepresentative.add(convertintArrToFloatArr(input[i]));
 }
 else
 {
 cluster[clusterId][0]+=1;
 int index=cluster[clusterId][0];
 cluster[clusterId][index]=i;

clusterRepresentative.set(clusterId,calculateClusterRepresentative(cluster[clu
sterId],input,
noOfTokens));
 }
 }
 for(int i=0;i {
 System.out.print("\n"+i+"\t");
 for(int j=1;j<=cluster[i][0];++j)
 {
 System.out.print(" "+cluster[i][j]);
 }
 }
 }
 private static Float[] convertintArrToFloatArr(int[] input)
 {
 int size=input.length;
 Float[] answer = new Float[size];
 for(int i=0;i {
```

```
 answer[i]=(float)input[i];
 }
 return answer;
 }
 private static float calculateSimilarity(Float[] a,Float[] b)
 {
 float answer=0;
 for(int i=0;i {
 answer+=a[i]*b[i];
 }
 return answer;
 }
 private static Float[] calculateClusterRepresentative(int[] cluster,int [][]
input,int noOFTokens)
 {
 Float[] answer= new Float[noOFTokens];
 for(int i=0;i {
 answer[i]=Float.parseFloat("0");
 }
 for(int i=1;i<=cluster[0];++i)
 {
 for(int j=0;j {
 answer[j]+=input[cluster[i]][j];
 }
 }
 for(int i=0;i {
 answer[i]/=cluster[0];
 }
 return answer;
 }
}
```

Output -

Enter the no of Tokens

5

Enter the no of Documents

5

Enter the threshhold

10

Enter the Document Token Matrix

Enter(0,0)

1

```
Enter(0,1)
3
Enter(0,2)
3
Enter(0,3)
2
Enter(0,4)
2
Enter(1,0)
2
Enter(1,1)
1
Enter(1,2)
0
Enter(1,3)
1
Enter(1,4)
2
Enter(2,0)
0
Enter(2,1)
2
Enter(2,2)
0
Enter(2,3)
0
Enter(2,4)
1
Enter(3,0)
0
Enter(3,1)
```

```
3
Enter(3,2)
1
Enter(3,3)
0
Enter(3,4)
5
Enter(4,0)
1
Enter(4,1 1)
0
Enter(4,2)
1
Enter(4,3)
0
Enter(4,4)
1
0 0 1 3
1 2
2 4
```