|  |  |  |
| --- | --- | --- |
| Sukkur_IBA_New_Logo | **Sukkur IBA University**  **Department of Computer Science** | **C:\Users\Saif Hassan\Downloads\CS logo (3).jpg** |

**DATA STRUCTURES**

**Lab01 – Arrays, LinkesLists**

**Instructor: Saif Hassan**

**READ IT FIRST**

Prior to start solving the problems in this assignments, please give full concentration on following points.

1. WORKING – This is individual lab. If you are stuck in a problem contact your teacher, but, in mean time start doing next question (don’t waste time).
2. DEADLINE – 11th March, 2022
3. SUBMISSION – This assignment needs to be submitted in a soft copy.
4. WHERE TO SUBMIT – Please visit your LMS.
5. WHAT TO SUBMIT – Submit this docx and pdf file.

**KEEP IT WITH YOU!**

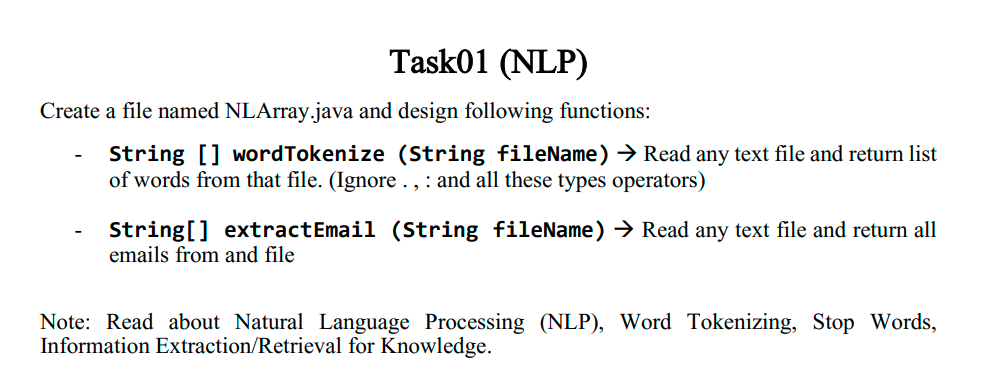
1. Indent your code inside the classes and functions. It’s a good practice!
2. It is not bad if you keep your code indented inside the loops, if and else blocks as well.
3. Comment your code, where it is necessary.
4. Read the entire question. Don’t jump to the formula directly.

I, \_\_**Amjad Ali**\_\_ with student ID \_**191-21-0001**\_Section \_\_**’A’**\_\_hereby declare that I do understand the instructions above and follow them. This is

my own work.

**Exercises**

**Task1 Description**



Solution:

1. package com.company;
3. import java.io.File;
4. import java.io.FileNotFoundException;
5. import java.util.ArrayList;
6. import java.util.Arrays;
7. import java.util.Scanner;
9. public class NLArray {
10. public static ArrayList extractEmail (String filename)
11. {
12. ArrayList<String> words=new ArrayList<String>();
13. String Content="";
14. String word="";
15. int index=-1;
16. try {
18. File myObj = new File(filename);
19. Scanner myReader = new Scanner(myObj);
21. while (myReader.hasNextLine()) {
22. Content=Content+myReader.nextLine();
23. }
24. myReader.close();
25. }catch (Exception e) {
26. System.out.println("File Not Found");
27. String[] arr3={"Wrong","Directory"};
28. return (ArrayList) Arrays.asList(arr3);
29. }
31. if(Content.charAt(Content.length()-1)!=' ')
32. Content=Content+" ";
34. for(int i=0;i<Content.length();i++)
35. {
37. boolean con=true;
38. if(Content.charAt(i)==' ' && word!="")
39. {
41. int length=word.length();
42. int at=word.indexOf('@');
43. int dot=-1;
44. if(at!=-1) {
45. for (int z = at; z < length; z++) {
46. if (word.charAt(z) == '.')
47. dot = z;
49. }
51. if (dot == -1) {
52. con = false;
53. } else {
54. if (at < 3)
55. con = false;
56. if (dot + 3 > length)
57. con = false;
58. if (at + 4 > dot)
59. con = false;
60. }
61. }
62. else
63. {
64. con=false;
65. }
66. if(con)
67. words.add(word);
68. word="";
69. }
70. else{
72. word=word+Content.charAt(i);

75. }

78. }
79. return words;
81. }
83. public static String[] wordTokenize (String filename)
84. {
85. String Content="";
86. try {
88. File myObj = new File(filename);
89. Scanner myReader = new Scanner(myObj);
91. while (myReader.hasNextLine()) {
92. Content=Content+myReader.nextLine();
93. }
94. myReader.close();
95. }catch (Exception e) {
96. System.out.println("File Not Found");
97. String[] arr={"Wrong","Directory"};
98. return arr;
99. }
101. if(Content.charAt(Content.length()-1)!=' ')
102. Content=Content+" ";
104. int size=0;
105. for(int i=0;i<Content.length();i++)
106. {
107. if(Content.charAt(i)==' ')
108. size++;
109. }
110. String Words[]=new String[size];
111. String word="";
112. int index=-1;
114. for(int i=0;i<Content.length();i++)
115. {

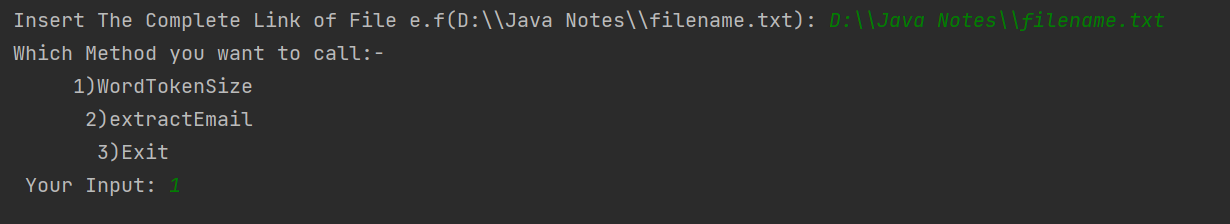
118. if(Content.charAt(i)==' ')
119. {
121. Words[++index]=word;
122. word="";
123. }
124. else{
126. word=word+Content.charAt(i);

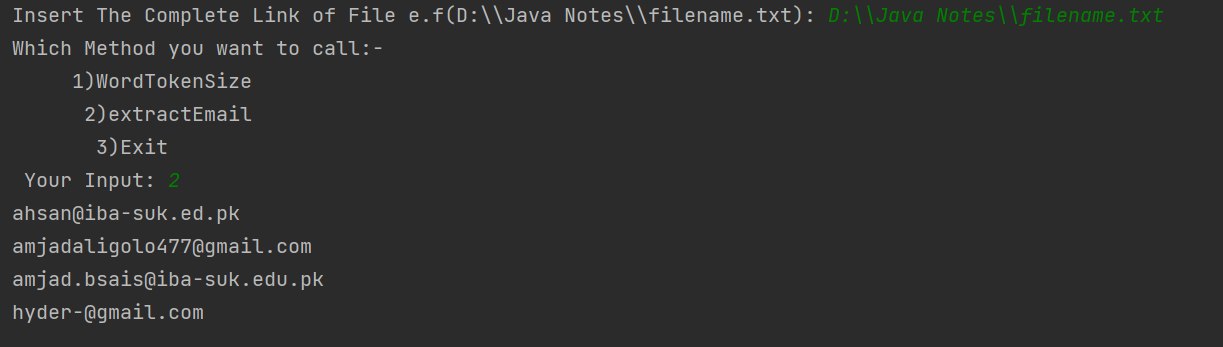
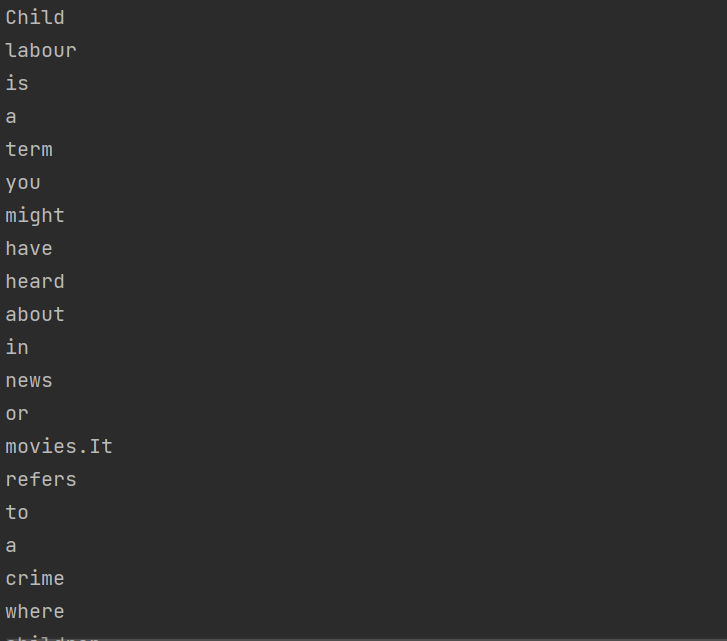
129. }

132. }
133. return Words;
135. }
137. //"D:\\Java Notes\\filename.txt"
138. public static void main(String[] args) {
139. Scanner sc=new Scanner(System.in);
140. System.out.print("Insert The Complete Link of File e.f(D:\\\\Java Notes\\\\filename.txt): ");
141. String link =sc.nextLine();
142. String[] Words=wordTokenize(link);
143. ArrayList<String> Words2=extractEmail(link);
144. System.out.print("Which Method you want to call:- \n 1)WordTokenSize\n 2)extractEmail\n 3)Exit\n Your Input: ");
145. int input =sc.nextInt();
146. if(input==1)
147. {
148. for(String word:Words)
149. System.out.println(word);
150. }
151. else if(input==2)
152. {
153. for(int i=0;i<Words2.size();i++)
154. System.out.println(Words2.get(i));
155. }
156. else if(input==3)
157. {
158. System.out.println("Have A Good Day");
159. System.exit(0);
160. }
161. else{
163. System.out.println("Wrong Input ");
164. System.exit(0);
166. }

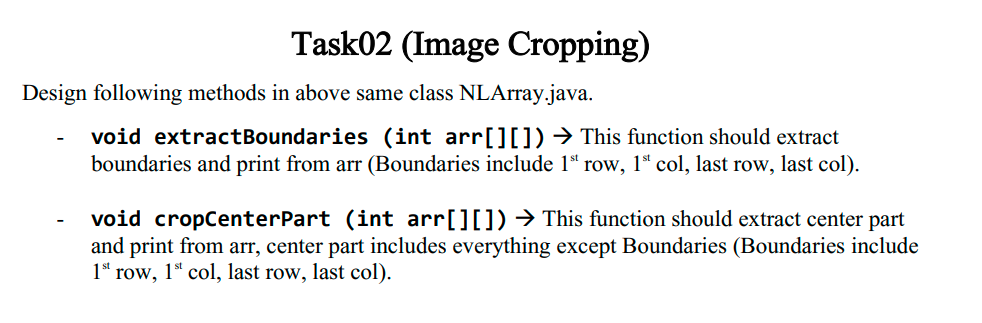
169. }
171. }

**Sample Input:**



**Sample Output**

**Task2 Description**



Solution:

1. package com.company;
3. import java.util.Random;
4. import java.util.Scanner;



9. public class NLArray2 {
10. public static boolean NConRep (int arr[][])
11. {
12. boolean cond=true;
13. for(int i=0;i< arr.length;i++)
14. {
15. for(int j=0;j<arr[0].length;j++)
16. {
17. if(arr[i][0]!=arr[i][j])
18. cond=false;
20. }
21. if(cond)
22. return true;
24. cond=true;
25. }

28. return false;
29. }
30. public static void extractBoundaries (int arr[][])
31. {
32. int frow[]=new int[arr[0].length];
33. int lrow[]=new int [arr[0].length];
34. int fcol[]=new int[arr.length-2];
35. int lcol[]=new int[arr.length-2];
36. if(arr.length<=2 || arr[0].length<=2)
37. {
38. System.out.println(" \n Output will be:- \n");
39. for(int i=0;i<arr.length;i++)
40. {
41. for(int j=0;j<arr[0].length;j++)
42. {
43. System.out.print(arr[i][j]+" ");
44. }
45. System.out.println();
46. }
48. }
49. else
50. {
51. System.out.println(" \n Output will be:- \n");
53. for(int i=0;i<arr[0].length;i++)
54. {
55. frow[i]=arr[0][i];
56. lrow[i]=arr[arr.length-1][i];
58. }
59. int index1=-1;
60. int index2=-1;
61. for(int i=1;i<arr.length-1;i++)
62. {
63. fcol[++index1]=arr[i][0];
64. lcol[++index2]=arr[i][arr[0].length-1];
65. }
67. for(int ele:frow)
68. System.out.print(ele+" ");
70. System.out.println();
71. for(int i=0;i<fcol.length;i++)
72. {
73. System.out.print(fcol[i]);
74. for(int s=0;s<arr[0].length-2;s++)
75. System.out.print(" ");
76. //if(arr[0].length!=3)
77. System.out.print(" ");
79. System.out.print(lcol[i]);
80. System.out.println();
81. }
83. for(int ele:lrow)
84. System.out.print(ele+" ");
86. }

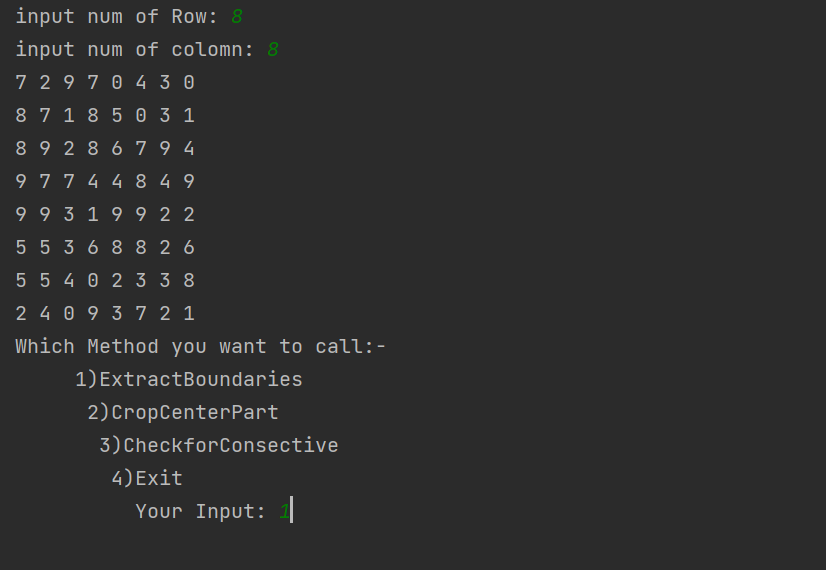
89. }
91. static void cropCenterPart(int arr[][])
92. {
93. if(arr.length<=2 || arr[0].length<=2)
94. {
95. System.out.println("The Given Array Can't be processed");
96. }
97. else
98. {
99. System.out.println(" \n Output will be:- \n");
100. for(int i=1;i<arr.length-1;i++)
101. {
102. System.out.print(" ");
103. for(int j=1;j<arr[0].length-1;j++)
104. {
105. System.out.print(arr[i][j]+" ");
106. }
107. System.out.println();
108. }
109. }
110. }
112. public static void main(String[] args) {
113. Scanner sc =new Scanner(System.in);
114. Random rand=new Random();
115. System.out.print("input num of Row: ");
116. int row=sc.nextInt();
117. System.out.print("input num of colomn: ");
118. int colomn=sc.nextInt();
119. int[][] arr=new int[row][colomn];
120. for(int i=0;i<row;i++)
121. {
123. for(int j=0;j<colomn;j++)
124. {
125. int a= rand.nextInt(10);
127. arr[i][j]=a;
128. System.out.print(arr[i][j]+" ");
129. }
130. System.out.println();
131. }
133. System.out.print("Which Method you want to call:- \n 1)ExtractBoundaries\n 2)CropCenterPart\n 3)CheckforConsective\n 4)Exit\n Your Input: ");
134. int input =sc.nextInt();
135. if(input==1)
136. {
137. extractBoundaries(arr);
138. }
139. else if(input==2)
140. {
141. cropCenterPart(arr);
142. }
143. else if(input==3)
144. {
145. System.out.print("Result = ");
146. System.out.println(NConRep(arr));
147. }
148. else if(input==4)
149. {
150. System.out.println("Have A Good Day");
151. System.exit(0);
152. }
153. else {
155. System.out.println("Wrong Input ");
156. System.exit(0);
158. }



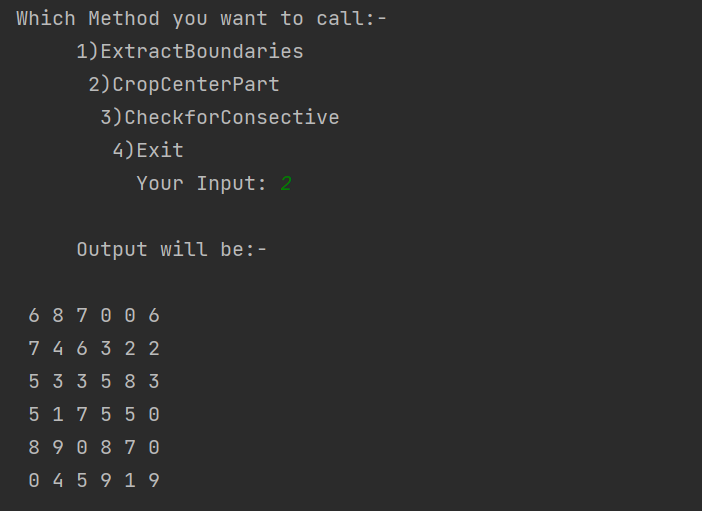
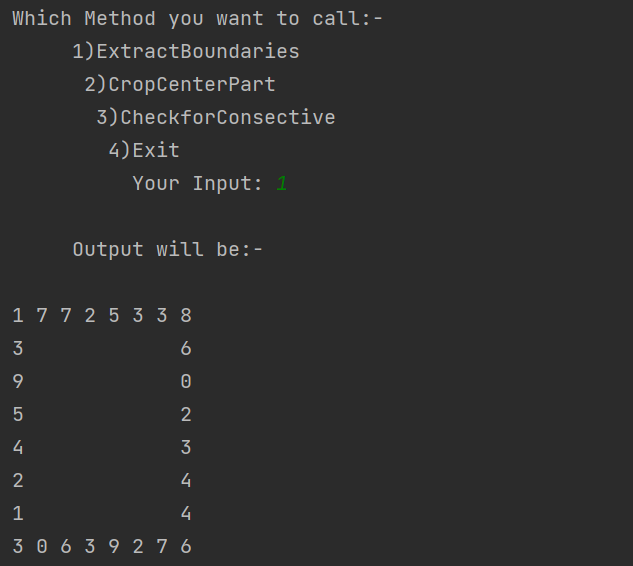



166. }
167. }

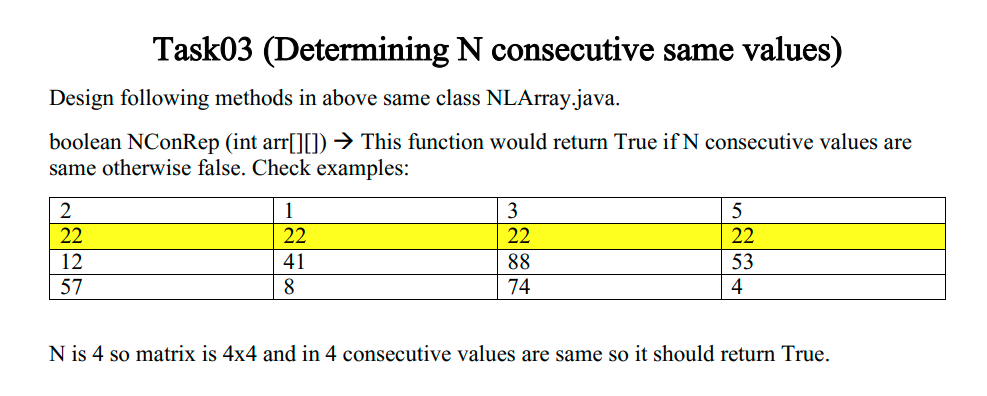
**Sample Input:**

****

**Sample Output**



**Task3 Description**

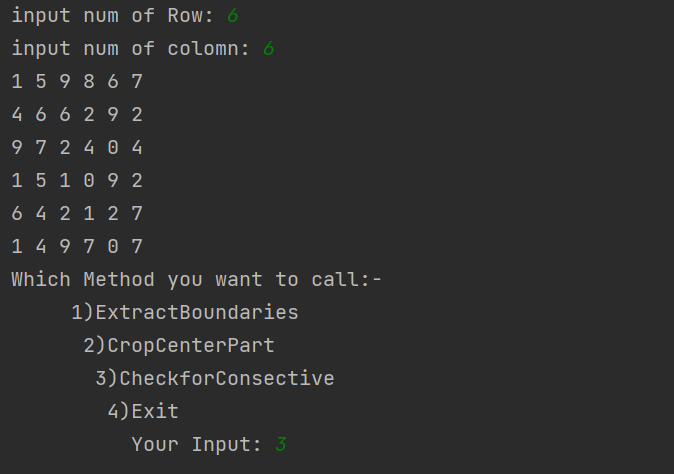


Solution:

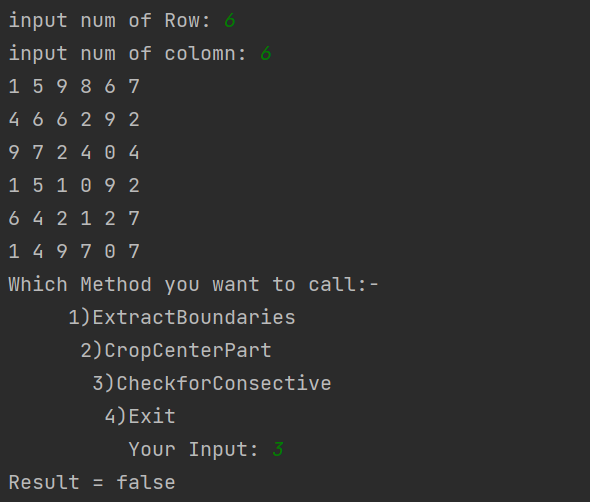
1. public static boolean NConRep (int arr[][])
2. {
3. boolean cond=true;
4. for(int i=0;i< arr.length;i++)
5. {
6. for(int j=0;j<arr[0].length;j++)
7. {
8. if(arr[i][0]!=arr[i][j])
9. cond=false;
11. }
12. if(cond)
13. return true;
15. cond=true;
16. }

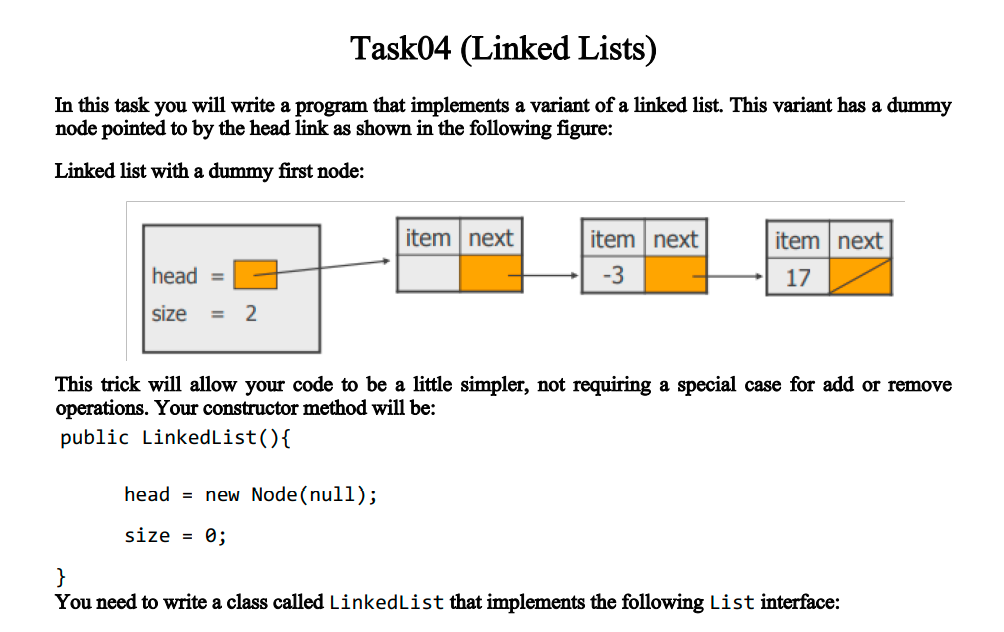
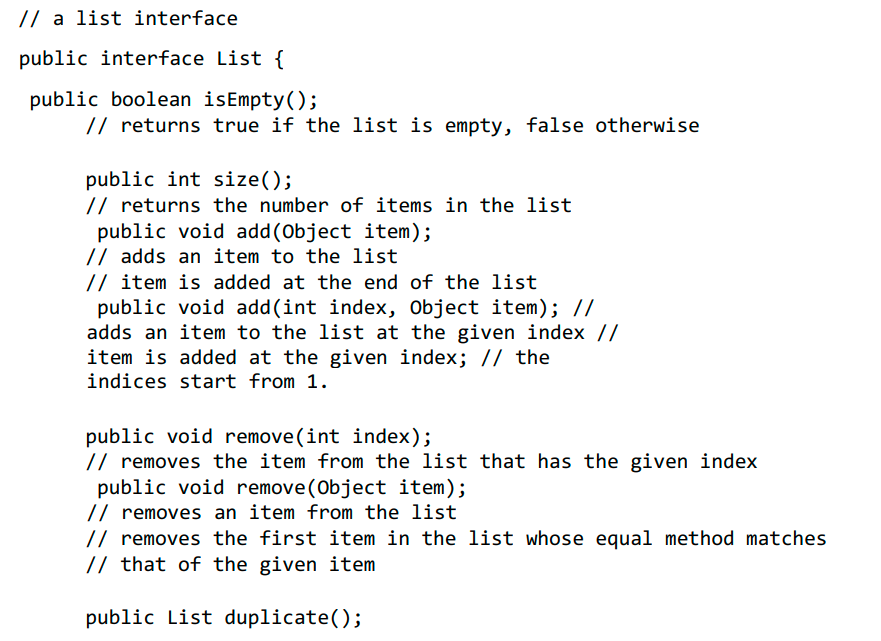
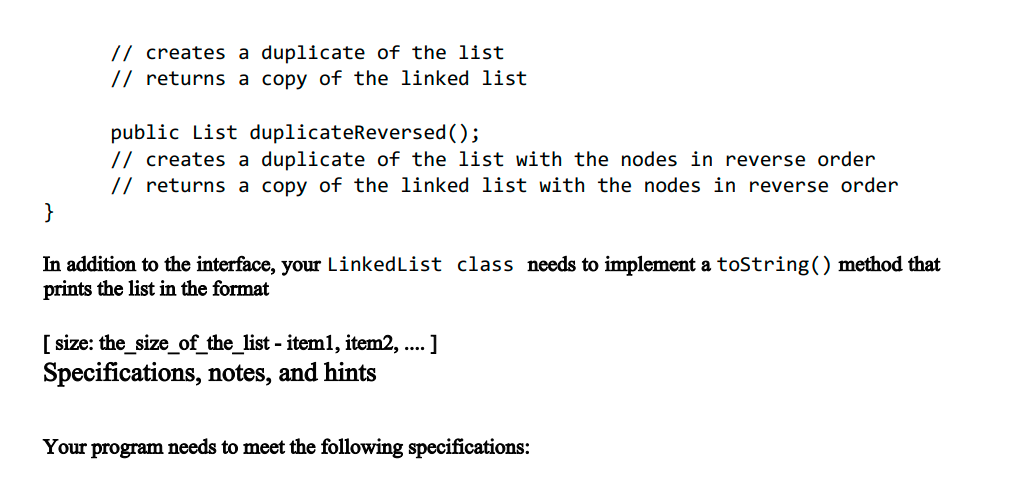
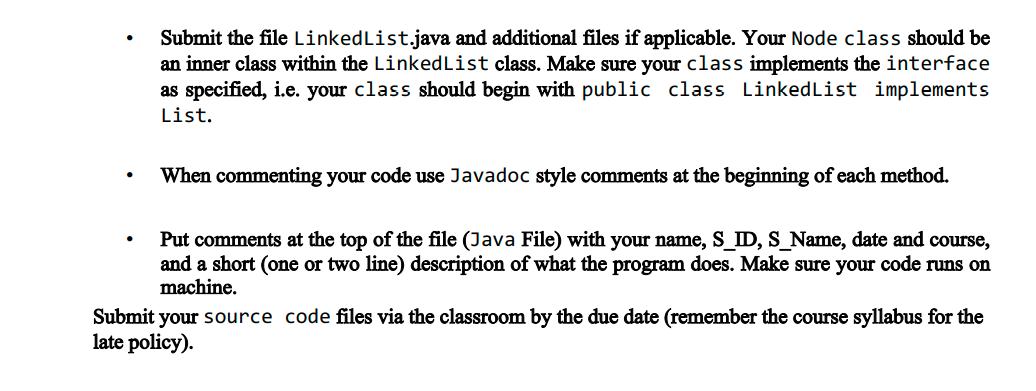
19. return false;
20. }

**Sample Input:**

****

**Sample Output**



   **Solution:**

1. **List Interface**
2. package com.company;
3. public interface List {
5. public boolean isEmpty();
6. // returns true if the list is empty, false otherwise

9. public int size();
10. // returns the number of items in the list

13. public void add(int item);
14. // adds an item to the list
15. // item is added at the end of the list

18. public void add(int index, int item);
19. // adds an item to the list at the given index
20. // item is added at the given index;
21. // the indices start from 1.
23. public void removeIndex(int index);
24. // removes the item from the list that has the given index
26. public void remove(int item);
27. // removes an item from the list
28. // removes the first item in the list whose equal method matches that of the given item
30. public List duplicate();
31. // creates a duplicate of the list
32. // returns a copy of the linked list
34. public List duplicateReversed();
35. // creates a duplicate of the list with the nodes in reverse order
36. // returns a copy of the linked list with the nodes in reverse order
38. }

* 1. **Linked List Class**

1. package com.company;

4. public class Linked\_List implements List{
6. private int size=0;
7. private Node Head;

10. public class Node{
11. int data;
12. Node next;
14. Node(int data)
15. {
16. this.data=data;
18. }
19. }
20. public void incSize(){
21. size++;
22. }

25. public boolean isEmpty()
26. {
27. return Head==null;
28. }
30. public void add(int data)
31. {
32. Node newNode=new Node(data);
33. if(isEmpty())
34. {
35. Head=newNode;
36. }
37. else{
38. newNode.next=Head;
39. Head =newNode;
40. /\* Node Current=Head;
41. while(Current.next!=null)
42. {
43. Current=Current.next;
44. }
45. Current.next=newNode;\*/
47. }
48. size++;
50. }
52. @Override
53. public int size() {
55. return size;
56. }
58. @Override
59. public void add(int index, int item) {
61. if(index>size)
62. System.out.println("index Out of Bound");
63. else{
64. int i=1;
65. Node current=Head;
66. while(i<index)
67. {
68. current=current.next;
69. i++;
70. }
71. current.data=item;
72. System.out.println("Successfully Added");
73. }
74. }
76. @Override
77. public void removeIndex(int index) {
78. if(size<index)
79. System.out.println("Index out of bound");
80. else if(index==1)
81. {
82. Head=Head.next;
83. }
84. else{
85. int i=1;
86. Node current=Head;
87. while(i+1!=index)
88. {
89. current=current.next;
90. i++;
91. }
92. current.next=current.next.next;
93. System.out.println("Successfully Removed");
94. size--;
95. }
97. }
99. @Override
100. public void remove(int item) {
101. if (size == 0) {
102. System.out.println("List Is Empty.");
103. } else {
104. boolean cond = false;
105. Node Current = Head;
106. Node oneBackCurrent = Head;
107. while (Current.next != null) {
109. if (Current.data==item) {
110. cond = true;
111. break;
113. }
114. oneBackCurrent = Current;
115. Current = Current.next;

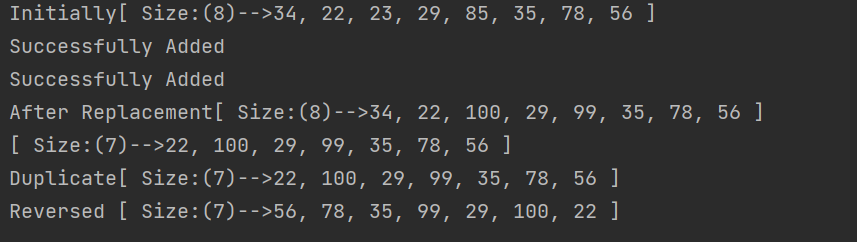

119. }
120. if (Current.data==item)
121. cond = true;
123. if(Head.data==item)
124. {
125. Head=Head.next;
126. size--;
127. }
128. else if (cond) {
129. oneBackCurrent.next= oneBackCurrent.next.next;
130. System.out.println("Succesfully Removed");
131. size--;
132. } else {
133. System.out.println("No Such Element In the List");
134. }
135. }
136. }
137. public List duplicateReversed()
138. {
139. Linked\_List list=new Linked\_List();
140. Node Current=Head;
141. for(int i=0;i<size;i++)
142. {
144. list.add(Current.data);
145. Current=Current.next;
146. }

149. return list;
151. }
153. public List duplicate()
154. {
155. Linked\_List list=new Linked\_List();
156. Node Current=Head;
157. for(int i=0;i<size;i++)
158. {
159. Node newNode=new Node(Current.data);
161. if(list.isEmpty()) {
162. list.Head = newNode;
163. }
164. else{
165. Node Check=list.Head;
166. while(Check.next!=null)
167. {
168. Check=Check.next;
169. }
170. Check.next=newNode;

173. }
174. list.incSize();
175. Current=Current.next;
177. }
179. return list;
180. }
182. public void print()
183. {
184. if(isEmpty())
185. {
186. System.out.println("List is Empty");
188. }
189. else{
190. Node Current=Head;
191. while(Current.next!=null)
192. {
193. System.out.print(Current.data+", ");
194. Current=Current.next;
195. }
196. System.out.println(Current.data);
198. }
199. }
201. public String toString()
202. {
203. String Str ="[ Size:("+size+")-->";
204. Node Current=Head;
205. while(Current.next!=null)
206. {
207. Str+=Current.data+", ";
208. Current=Current.next;
209. }
210. Str+=Current.data+" ]";
212. return Str;
213. }
215. }
216. **Linked List Demo Class**
217. package com.company;
219. public class Linked\_ListDemo {
220. public static void main(String[] args) {
221. Linked\_List list=new Linked\_List();
222. list.add(56);
223. list.add(78);
224. list.add(35);
225. list.add(85);
226. list.add(29);
227. list.add(23);
228. list.add(22);
229. list.add(34);
230. System.out.println("Initially"+list);
231. list.add(3,100);
232. list.add(5,99);
233. System.out.println("After Replacement"+list);
234. List list1,list2;
235. list.remove(34);
236. System.out.println(list);
237. list1=list.duplicate();
238. list2=list.duplicateReversed();
239. System.out.println("Duplicate"+list1);
240. System.out.println("Reversed "+list2);


244. }
245. }

**Sample Input:**

**Sample Output**