**SET A**

**QUESTION 1**

1 import java.util.\*;  
 2 class Node{  
 3 int data;  
 4 Node next;  
 5 Node(){  
 6 this.data = 0;  
 7 this.next = null;  
 8 }  
 9 }  
 10 public class singly\_LinketList{  
 11 static Node head;  
 12 public static void main(String args[]){  
 13 Scanner sc = new Scanner(System.in);  
 14 Node n1 = new Node();  
 15 Node n2 = new Node();  
 16 Node n3 = new Node();  
 17 n1.data = 5;  
 18 n2.data = 6;  
 19 n3.data = 7;  
 20 head = n1;  
 21 n1.next = n2;   
 22 n2.next = n3;  
 23 while(true){  
 24 System.out.println("\nEnter 1 to add a new Node at the beginning");  
 25 System.out.println("Enter 2 to add a new Node at the end");  
 26 System.out.println("Enter 3 to add a new Node at a specific location");  
 27 System.out.println("Enter 4 to delete a Node at the beginning");  
 28 System.out.println("Enter 5 to delete a Node at the end");  
 29 System.out.println("Enter 6 to delete a Node at a specific location");  
 30 System.out.println("Enter 7 to print all the nodes");  
 31 System.out.println("Enter 8 to exit");  
 32 System.out.print("Enter your choice : ");  
 33 int input = sc.nextInt();  
 34 if(input==1){  
 35 System.out.print("\nEnter the data of Node to be added at the beginning : ");  
 36 int data\_ = sc.nextInt();  
 37 node\_beginning(data\_);  
 38 }  
 39 else if(input==2){  
 40 System.out.print("\nEnter the data of Node to be added at the end : ");  
 41 int data\_ = sc.nextInt();  
 42 node\_end(data\_);  
 43 }  
 44 else if(input==3){  
 45 System.out.print("\nEnter the data of Node : ");  
 46 int data\_ = sc.nextInt();  
 47 System.out.print("\nEnter the index at which Node is to be added : ");  
 48 int index = sc.nextInt();  
 49 add\_node\_index(index,data\_);  
 50   
 51 }  
 52 else if(input==4){  
 53 System.out.print("\nDeleting the node at the beginning");  
 54 del\_first();  
 55 }  
 56 else if(input==5){  
 57 del\_last();  
 58 }  
 59 else if(input==6){  
 60 System.out.print("\nEnter the index of node to be deleted ");  
 61 int index = sc.nextInt();  
 62 del\_index(index);  
 63 }  
 64 else if(input==7){  
 65 print\_nodes();   
 66 }  
 67 else if(input == 8){  
 68 System.out.print("Exiting loop");  
 69 break;  
 70 }  
 71   
 72 else  
 73 System.out.print("\nWrong Input!! Enter again ");  
 74 }  
 75   
 76 }  
 77   
 78 //Adding node at the beginning  
 79 public static void node\_beginning(int data\_){  
 80 Node new\_node = new Node();  
 81 new\_node.data = data\_;  
 82 new\_node.next = head;  
 83 head = new\_node;  
 84 }  
 85 //Adding node at the end   
 86 public static void node\_end(int data\_){  
 87 Node new\_node = new Node();  
 88 new\_node.data = data\_;  
 89 Node temp = new Node();  
 90 temp = head;  
 91 if(head==null){  
 92 head = new\_node;  
 93 }  
 94 else{  
 95 while(temp.next!=null){  
 96 temp = temp.next;  
 97 }  
 98 temp.next = new\_node;   
 99 }   
100 }  
101   
102 //Print all the nodes  
103 public static void print\_nodes(){  
104 Node temp = new Node();  
105 temp = head;  
106 if(temp == null){  
107 System.out.print("\nNo Elements found!\n");  
108 }  
109 else{  
110 while(temp.next!=null){  
111 System.out.print(temp.data+"--->");  
112 temp = temp.next;  
113 }  
114 System.out.print(temp.data+"\n");  
115 }  
116 }  
117   
118 //Adding node at a specific index  
119 public static void add\_node\_index(int index,int data\_){   
120 Node temp = new Node();  
121 Node new\_node = new Node();  
122 new\_node.data = data\_;  
123 int ctr = 0;  
124 temp = head;  
125 if(index == 0 ){  
126 new\_node.next = head;  
127 head = new\_node;  
128 ctr = 1;  
129 }  
130 else if(index == 1){  
131 new\_node.next = head.next;  
132 head.next = new\_node;  
133 ctr = 1;  
134 }  
135   
136 else {  
137 for(int i = 0 ; i<index-1 ; i++){  
138 if(temp.next==null){  
139 System.out.println("Entered index is out of bound.");  
140 ctr = 1;  
141 break;  
142 }  
143 else  
144 temp =temp.next;  
145 }  
146 if(ctr==0){  
147 new\_node.next = temp.next;  
148 temp.next=new\_node;  
149 }  
150 }  
151   
152   
153 }  
154   
155 //Deleting the node at the beginning  
156 public static void del\_first(){  
157 Node temp = new Node();  
158 temp = head;  
159 if(head!= null){  
160 head = temp.next;  
161 temp.next = null;  
162 }  
163 else  
164 System.out.print("\nList in empty");   
165 }  
166   
167 //Deleting the last element  
168 public static void del\_last(){  
169 Node temp = new Node();  
170 temp = head;  
171 if(head==null){  
172 System.out.print("\nNo elements in the list");  
173 }  
174 else if(head.next==null){  
175 head=null;  
176 }  
177 else{  
178 while(3>1){  
179 if(temp.next == null){  
180 temp = temp.next;  
181 break;  
182 }   
183 if((temp.next).next == null){  
184 temp.next = null;  
185 break;  
186 }  
187   
188 temp = temp.next;   
189 }  
190 }  
191 }  
192   
193 //Deleting alternate elements  
194 public static void del\_alt(){  
195 Node temp = new Node();  
196 temp = head;  
197 int x = 0;  
198 if(head==null){  
199 System.out.print("\nThe List is empty");  
200 }  
201 else if(head.next==null){  
202 System.out.print("\nList contains only one node");  
203 }  
204 else{  
205 while(temp.next!=null){  
206 if(temp.next.next==null){  
207 temp.next=null;  
208 }  
209 else{  
210 temp.next=temp.next.next;  
211 temp = temp.next;  
212 }  
213 }  
214 }  
215 }  
216   
217 //Deleting duplicates from unsorted list  
218 public static void del\_duplicate(){  
219 Node temp = new Node();  
220 temp = head;  
221 int x = 0;  
222 if(head==null){  
223 System.out.print("\nList is empty");  
224 }  
225 else if(head.next==null){  
226 System.out.print("\nList Contains just one Node");  
227   
228 }  
229 else{  
230 while(temp.next!=null && temp!=null){  
231 x = temp.data;  
232 Node temp2 = new Node();  
233 temp2 = temp;  
234 while(temp2.next!=null){  
235 if(x==temp2.next.data){  
236 temp2.next = temp2.next.next;  
237 }  
238   
239 else{  
240 temp2= temp2.next;   
241 }  
242   
243 }  
244 temp = temp.next;  
245 }  
246   
247 }  
248   
249 }  
250   
251 //Lenght  
252 public static int lastIndex\_node(Node n){  
253 int ctr = 0;  
254 while(n!=null){  
255 ctr = ctr+1;  
256 n=n.next;  
257 }  
258   
259 return ctr-1;  
260 }  
261   
262 //Getting the position first and last occurence of node  
263 public static void occur\_f(int s){  
264 Node temp = new Node();  
265 temp = head;  
266 ArrayList<Integer> list = new ArrayList<Integer>();  
267 if(head==null){  
268 System.out.print("\nList is empty");  
269 }  
270 else if(head.next==null){  
271 if(head.data==s){  
272 System.out.print("\nOnly one Node present at Position 1");  
273 }  
274 else  
275 System.out.print("No occurence of "+s+" found");  
276 }  
277 else{  
278 int ctr = 1;  
279 int x = 0;  
280 while(temp.next!=null){  
281 if(s==temp.data){  
282 list.add(ctr);  
283 x=1;  
284 }  
285 ctr=ctr+1;  
286 temp=temp.next;  
287 }  
288 if(s==temp.data){  
289 list.add(ctr);  
290 x=1;  
291 }  
292 if(x==0){  
293 System.out.print("\nNo Occurence found");  
294 }  
295 if(list.size()==1){  
296 System.out.print("\nThe only occurence of "+s+" is at position :"+list.get(0)+"\n");  
297 }  
298 else if(list.size()>1){  
299 System.out.print("\nThe first occurence of "+s+" is at position :"+list.get(0)+"\n");  
300 System.out.print("\nThe last occurence of "+s+" is at position :"+list.get(list.size()-1)+"\n\n");  
301 }  
302   
303 }  
304   
305 }  
306   
307 //SWAPPING 1ST AND LAST ELEMENT  
308 public static void swap(){  
309 Node temp = new Node();  
310 temp = head;  
311 if(head==null){  
312 System.out.print("\nList is empty");  
313 }  
314 else if(head.next==null){  
315 System.out.print("List contains only one element.");  
316 }  
317 else{  
318 while(temp.next!=null){  
319 temp=temp.next;  
320 }  
321 int num = temp.data;  
322 temp.data = head.data;  
323 head.data = num;  
324 }  
325 }  
326   
327 //PAIRWISE SWAPPING OF NODES  
328 public static void swap\_pair(){  
329 Node temp = new Node();  
330 temp = head;  
331 int y = head.data;  
332 if(head==null){  
333 System.out.print("\nList is empty");  
334 }  
335 else if(head.next==null){  
336 System.out.print("List contains only one element.");  
337 }  
338 else{  
339 temp.data = temp.next.data;  
340 temp.next.data = y;  
341 temp = temp.next.next;  
342   
343 while(temp.next!=null){  
344 int x = temp.data;  
345 temp.data = temp.next.data;  
346 temp.next.data = x;  
347 if(temp.next.next!=null){  
348 temp = temp.next.next;  
349 }  
350 else if(temp.next.next==null){  
351 break;  
352 }   
353   
354 }  
355   
356 }  
357 }  
358   
359 //Deleting the element at a given index  
360 public static void del\_index(int index){  
361 Node temp = new Node();  
362 Node temp2 = new Node();  
363 temp = head;  
364 if(index<= lastIndex\_node(temp) && index>=0){  
365 if(index==0){  
366 head = temp.next;  
367 temp.next = null;  
368 }  
369 if(index==1){  
370 temp = temp.next;  
371 head.next = temp.next;  
372   
373 }  
374 else{  
375 for(int i = 0; i<=index-2 ; i++){  
376 temp = temp.next;  
377 if(i+2 == index){  
378 temp2 = temp.next.next;  
379 temp.next = temp2;  
380   
381 }  
382 }  
383 }  
384 }  
385 else  
386 System.out.print("\nIndex out of bounds\n");  
387 }  
388   
389 }  
390   
391

**QUESTION 2**

1 import java.util.\*;  
 2 class Node{  
 3 int data;  
 4 Node next;  
 5 Node(){  
 6 this.data = 0;  
 7 this.next = null;  
 8 }  
 9 }  
 10 public class sll\_fn{  
 11 static Node head;  
 12 public static void main(String args[]){  
 13 while(3>1){  
 14 int c= 0;  
 15 char ch;  
 16 Scanner sc = new Scanner(System.in);  
 17 singly\_LinketList sll = new singly\_LinketList();  
 18 System.out.print("\nEnter the size of linked list ");  
 19 int s = sc.nextInt();  
 20 for(int i = 0; i<s ; i++){  
 21 System.out.print("\nEnetr element "+(i+1)+" : ");  
 22 int x = sc.nextInt();  
 23 node\_end(x);  
 24 }  
 25 print\_nodes();  
 26 swap();  
 27 print\_nodes();  
 28 swap\_pair();  
 29 print\_nodes();  
 30 System.out.print("\nEnter the number to get the firsta nd last occurence :");  
 31 int y = sc.nextInt();  
 32 occur\_f(y);  
 33 print\_nodes();  
 34 del\_duplicate();  
 35 print\_nodes();  
 36 del\_alt();  
 37 print\_nodes();  
 38 do{  
 39 c=0;  
 40 System.out.print("\nDo you want to quit?\nEnter Y or y for YES \nEnter N or n for NO ");  
 41 ch = sc.next().charAt(0);  
 42 if((ch!='Y' && ch!='y') && (ch!='N' && ch!='n')){  
 43 System.out.print("\nINVALID INPUT");  
 44 c=1;  
 45 }  
 46 }while(c==1);  
 47 if(ch=='Y' || ch=='y')  
 48 break;  
 49 }  
 50   
 51   
 52 }  
 53 //Printing the nodes  
 54 public static void print\_nodes(){  
 55 Node temp = new Node();  
 56 temp = head;  
 57 if(temp == null){  
 58 System.out.print("\nNo Elements found!\n");  
 59 }  
 60 else{  
 61 while(temp.next!=null){  
 62 System.out.print(temp.data+"--->");  
 63 temp = temp.next;  
 64 }  
 65 System.out.print(temp.data+"\n");  
 66 }  
 67 }  
 68   
 69 //Adding note at the end  
 70 public static void node\_end(int data\_){  
 71 Node new\_node = new Node();  
 72 new\_node.data = data\_;  
 73 Node temp = new Node();  
 74 temp = head;  
 75 if(head==null){  
 76 head = new\_node;  
 77 }  
 78 else{  
 79 while(temp.next!=null){  
 80 temp = temp.next;  
 81 }  
 82 temp.next = new\_node;   
 83 }   
 84 }  
 85   
 86 //SWAPPING 1ST AND LAST ELEMENT  
 87 public static void swap(){  
 88 System.out.print("\nSwapping 1st and Last Node \n");  
 89 Node temp = new Node();  
 90 temp = head;  
 91 if(head==null){  
 92 System.out.print("\nList is empty");  
 93 }  
 94 else if(head.next==null){  
 95 System.out.print("List contains only one element.");  
 96 }  
 97 else{  
 98 while(temp.next!=null){  
 99 temp=temp.next;  
100 }  
101 int num = temp.data;  
102 temp.data = head.data;  
103 head.data = num;  
104 }  
105 }  
106   
107 //PAIRWISE SWAPPING OF NODES  
108 public static void swap\_pair(){  
109 System.out.print("\nSwapping Pairwise \n");  
110 Node temp = new Node();  
111 temp = head;  
112 int y = head.data;  
113 if(head==null){  
114 System.out.print("\nList is empty");  
115 }  
116 else if(head.next==null){  
117 System.out.print("List contains only one element.");  
118 }  
119 else{  
120 temp.data = temp.next.data;  
121 temp.next.data = y;  
122 temp = temp.next.next;  
123   
124 while(temp.next!=null){  
125 int x = temp.data;  
126 temp.data = temp.next.data;  
127 temp.next.data = x;  
128 if(temp.next.next!=null){  
129 temp = temp.next.next;  
130 }  
131 else if(temp.next.next==null){  
132 break;  
133 }   
134   
135 }  
136   
137 }  
138 }  
139 //Getting the position first and last occurence of node  
140 public static void occur\_f(int s){  
141 System.out.print("\nGetting first and last occurence \n");  
142 Node temp = new Node();  
143 temp = head;  
144 ArrayList<Integer> list = new ArrayList<Integer>();  
145 if(head==null){  
146 System.out.print("\nList is empty");  
147 }  
148 else if(head.next==null){  
149 if(head.data==s){  
150 System.out.print("\nOnly one Node present at Position 1");  
151 }  
152 else  
153 System.out.print("No occurence of "+s+" found");  
154 }  
155 else{  
156 int ctr = 1;  
157 int x = 0;  
158 while(temp.next!=null){  
159 if(s==temp.data){  
160 list.add(ctr);  
161 x=1;  
162 }  
163 ctr=ctr+1;  
164 temp=temp.next;  
165 }  
166 if(s==temp.data){  
167 list.add(ctr);  
168 x=1;  
169 }  
170 if(x==0){  
171 System.out.print("\nNo Occurence found");  
172 }  
173 if(list.size()==1){  
174 System.out.print("\nThe only occurence of "+s+" is at position :"+list.get(0)+"\n");  
175 }  
176 else if(list.size()>1){  
177 System.out.print("\nThe first occurence of "+s+" is at position :"+list.get(0)+"\n");  
178 System.out.print("\nThe last occurence of "+s+" is at position :"+list.get(list.size()-1)+"\n\n");  
179 }  
180   
181 }  
182   
183 }  
184   
185 //Deleting duplicates from unsorted list  
186 public static void del\_duplicate(){  
187 System.out.print("\nDeleting duplicates from list \n");  
188 Node temp = new Node();  
189 temp = head;  
190 int x = 0;  
191 if(head==null){  
192 System.out.print("\nList is empty");  
193 }  
194 else if(head.next==null){  
195 System.out.print("\nList Contains just one Node");  
196   
197 }  
198 else{  
199 while(temp.next!=null && temp!=null){  
200 x = temp.data;  
201 Node temp2 = new Node();  
202 temp2 = temp;  
203 while(temp2.next!=null){  
204 if(x==temp2.next.data){  
205 temp2.next = temp2.next.next;  
206 }  
207   
208 else{  
209 temp2= temp2.next;   
210 }  
211   
212 }  
213 temp = temp.next;  
214 }  
215   
216 }  
217   
218 }  
219   
220   
221 //Deleting alternate elements  
222 public static void del\_alt(){  
223 System.out.print("\nDeleting alternate nodes \n");  
224 Node temp = new Node();  
225 temp = head;  
226 int x = 0;  
227 if(head==null){  
228 System.out.print("\nThe List is empty");  
229 }  
230 else if(head.next==null){  
231 System.out.print("\nList contains only one node");  
232 }  
233 else{  
234 while(temp.next!=null){  
235 if(temp.next.next==null){  
236 temp.next=null;  
237 }  
238 else{  
239 temp.next=temp.next.next;  
240 temp = temp.next;  
241 }  
242 }  
243 }  
244 }  
245   
246   
247 }

**QUESTION 3**

1 import java.util.\*;  
 2 public class link\_compare{  
 3 static Node head1;  
 4 static Node head2;  
 5 public static void main(String args[]){  
 6 Scanner sc = new Scanner(System.in);  
 7 int list1\_size =0;  
 8 int list2\_size =0;  
 9 System.out.print("Enter no. of elements of list1:");  
10 list1\_size = sc.nextInt();  
11 for(int i =0; i<list1\_size; i++){  
12 System.out.print("Enter "+(i+1)+" value: ");  
13 int a = sc.nextInt();  
14 node\_end(a,1);  
15 }  
16 System.out.print("Enter no. of elements of list2:");  
17 list2\_size = sc.nextInt();  
18 for(int i =0; i<list2\_size; i++){  
19 System.out.print("Enter "+(i+1)+" value: ");  
20 int a = sc.nextInt();  
21 node\_end(a,2);  
22 }  
23 findIdentical(list1\_size,list2\_size);  
24 }  
25 public static void node\_end(int a,int b){  
26 Node n = new Node();  
27 n.data = a;  
28 Node temp = new Node();  
29 if(b==1){  
30 temp = head1;  
31 if(head1 == null){  
32 head1 = n;  
33 }  
34 else{  
35 while(temp.next!=null){  
36 temp = temp.next;  
37 }  
38 temp.next = n;  
39 }  
40 }  
41 else if(b==2){  
42 temp = head2;  
43 if(head2 == null){  
44 head2 = n;  
45 }  
46 else{  
47 while(temp.next!=null){  
48 temp = temp.next;  
49 }  
50 temp.next = n;  
51 }  
52 }  
53 }  
54 public static void findIdentical(int list1\_size,int list2\_size){  
55 if(list1\_size != list2\_size){  
56 System.out.println("The two lists are NON-IDENTICAL");  
57 }  
58 else{  
59 Node temp1 = head1;  
60 Node temp2 = head2;  
61 int ctr = 0;  
62 while(temp1!= null && temp2!= null){  
63 if(temp1.data == temp2.data) ctr++;  
64 temp1 = temp1.next;  
65 temp2 = temp2.next;  
66 }  
67 if(ctr == list1\_size) System.out.println("The two lists are IDENTICAL");  
68 else System.out.println("The two lists are NON-IDENTICAL");  
69 }  
70 }  
71 }

**QUESTION 4**

1 import java.util.\*;  
 2 public class link\_Palin{  
 3 static Node head;  
 4 public static void main(String args[]){  
 5 Scanner sc = new Scanner(System.in);  
 6 System.out.print("\nEnter the size of linked list:");  
 7 int s = sc.nextInt();  
 8 int a[] = new int[s];  
 9 for(int i = 0; i<s ; i++){  
10 System.out.print("\nEnetr element "+(i+1)+" : ");  
11 a[i] = sc.nextInt();  
12 node\_end(a[i]);  
13 }  
14   
15   
16 System.out.print("\nThe entered Nodes are :");  
17 print\_nodes();  
18   
19 int p = check\_pal(a);  
20 if(p==1){  
21 System.out.print("\nThe list is palindrome");  
22 }  
23 else  
24 System.out.print("\nThe list is not palindrome");  
25   
26   
27 }  
28   
29 //Printing the nodes  
30 public static void print\_nodes(){  
31 Node temp = new Node();  
32 temp = head;  
33 if(temp == null){  
34 System.out.print("\nNo Elements found!\n");  
35 }  
36 else{  
37 while(temp.next!=null){  
38 System.out.print(temp.data+"--->");  
39 temp = temp.next;  
40 }  
41 System.out.print(temp.data+"\n");  
42 }  
43 }  
44   
45 //Adding node at the end  
46 public static void node\_end(int data\_){  
47 Node new\_node = new Node();  
48 new\_node.data = data\_;  
49 Node temp = new Node();  
50 temp = head;  
51 if(head==null){  
52 head = new\_node;  
53 }  
54 else{  
55 while(temp.next!=null){  
56 temp = temp.next;  
57 }  
58 temp.next = new\_node;   
59 }   
60 }  
61   
62 //Checking for palindrome  
63 public static int check\_pal(int[] b){   
64 int ctr = 0;  
65 int l = b.length;  
66 for (int j=0; j<l/2; j++) {  
67 if (b[j]==(b[l-j-1])){  
68 ctr = ctr+1;  
69 }  
70 else  
71 continue;   
72 }  
73 if(ctr==l/2){  
74 return 1;  
75 }   
76 else{  
77 return 0;  
78 }   
79 }  
80   
81 }

**QUESTION 5**

1 import java.util.\*;  
 2 public class Swap\_ListQ{  
 3 static Node head;  
 4 public static void main(String args[]){  
 5 Scanner sc = new Scanner(System.in);  
 6 System.out.print("\nEnter the size of linked list ");  
 7 int s = sc.nextInt();  
 8 for(int i = 0; i<s ; i++){  
 9 System.out.print("\nEnetr element "+(i+1)+" : ");  
10 int x = sc.nextInt();  
11 node\_end(x);  
12 }  
13 print\_nodes();  
14 swap\_pair();  
15 print\_nodes();  
16 }  
17   
18 public static void node\_end(int data\_){  
19 Node new\_node = new Node();  
20 new\_node.data = data\_;  
21 Node temp = new Node();  
22 temp = head;  
23 if(head==null){  
24 head = new\_node;  
25 }  
26 else{  
27 while(temp.next!=null){  
28 temp = temp.next;  
29 }  
30 temp.next = new\_node;   
31 }   
32 }  
33   
34 public static void print\_nodes(){  
35 Node temp = new Node();  
36 temp = head;  
37 if(temp == null){  
38 System.out.print("\nNo Elements found!\n");  
39 }  
40 else{  
41 while(temp.next!=null){  
42 System.out.print(temp.data+"--->");  
43 temp = temp.next;  
44 }  
45 System.out.print(temp.data+"\n");  
46 }  
47 }  
48   
49 public static void swap\_pair(){  
50 System.out.print("\nSwapping Pairwise \n");  
51 Node temp = new Node();  
52 temp = head;  
53 int y = head.data;  
54 if(head==null){  
55 System.out.print("\nList is empty");  
56 }  
57 else if(head.next==null){  
58 System.out.print("List contains only one element.");  
59 }  
60 else{  
61 temp.data = temp.next.data;  
62 temp.next.data = y;  
63 temp = temp.next.next;  
64   
65 while(temp.next!=null){  
66 int x = temp.data;  
67 temp.data = temp.next.data;  
68 temp.next.data = x;  
69 if(temp.next.next!=null){  
70 temp = temp.next.next;  
71 }  
72 else if(temp.next.next==null){  
73 break;  
74 }   
75   
76 }  
77   
78 }  
79 }  
80   
81 }   
82

**SET B**

**QUESTION 6**

1 import java.util.\*;  
 2 public class Linked\_SetB\_Q6{  
 3 static Node head;  
 4 public static void main(String args[]){  
 5 Scanner sc = new Scanner(System.in);  
 6 System.out.print("\nEnter the number : ");  
 7 String input = sc.next();  
 8 int ctr = 0;  
 9 for(int i = 0; i < input.length(); i++){  
10 if(Character.isDigit(input.charAt(i)) == true){   
11 ctr = ctr+1;  
12 }  
13 else  
14 continue;  
15 }  
16 if(ctr==input.length()){  
17 int num = Integer.parseInt(input);  
18 convert\_Binary(num);  
19 print\_nodes();  
20 }   
21 else{  
22 System.out.print("\nInput is not an Integer");  
23 }  
24 }  
25   
26   
27 //Printing the nodes  
28 public static void print\_nodes(){  
29 Node temp = new Node();  
30 temp = head;  
31 if(temp == null){  
32 System.out.print("\nNo Elements found!\n");  
33 }  
34 else{  
35 while(temp.next!=null){  
36 System.out.print(temp.data+"--->");  
37 temp = temp.next;  
38 }  
39 System.out.print(temp.data+"\n");  
40 }  
41 }  
42   
43 //Adding node at the end  
44 public static void node\_end(int data\_){  
45 Node new\_node = new Node();  
46 new\_node.data = data\_;  
47 Node temp = new Node();  
48 temp = head;  
49 if(head==null){  
50 head = new\_node;  
51 }  
52 else{  
53 while(temp.next!=null){  
54 temp = temp.next;  
55 }  
56 temp.next = new\_node;   
57 }   
58 }  
59   
60 //Binary Conversion  
61 public static void convert\_Binary(int num){  
62 int n;  
63 while(num!=1){  
64 n=num%2;  
65 num = num/2;  
66 node\_end(n);  
67 }  
68 node\_end(1);  
69 }  
70   
71 }

**QUESTION 7**

1 import java.util.\*;  
 2 public class Linked\_SetB\_Q7{  
 3 static Node head1;  
 4 static Node head2;  
 5 static Node head3;  
 6 public static void main(String args[]){  
 7 Scanner sc = new Scanner(System.in);  
 8 System.out.print("Enter size of lists: ");  
 9 int size = sc.nextInt();  
10 System.out.println("Enter elements of list 1.");  
11 for(int i =0; i<size; i++){  
12 System.out.print("Enter "+(i+1)+" value: ");  
13 int a = sc.nextInt();  
14 node\_end(a,1);  
15 }  
16 System.out.println("Enter elements of list 2.");  
17 for(int i =0; i<size; i++){  
18 System.out.print("Enter "+(i+1)+" value: ");  
19 int a = sc.nextInt();  
20 node\_end(a,2);  
21 }  
22 shuffleMerge(size);  
23 print\_nodes();  
24 }  
25 public static void node\_end(int a,int b){  
26 Node n = new Node();  
27 n.data = a;  
28 Node temp = new Node();  
29 if(b==1){  
30 temp = head1;  
31 if(head1 == null){  
32 head1 = n;  
33 }  
34 else{  
35 while(temp.next!=null){  
36 temp = temp.next;  
37 }  
38 temp.next = n;  
39 }  
40 }  
41 else if(b==2){  
42 temp = head2;  
43 if(head2 == null){  
44 head2 = n;  
45 }  
46 else{  
47 while(temp.next!=null){  
48 temp = temp.next;  
49 }  
50 temp.next = n;  
51 }  
52 }  
53 }  
54 public static void node\_end(int a){  
55 Node n = new Node();  
56 n.data = a;  
57 Node temp = new Node();  
58 temp = head3;  
59 if(head3 == null){  
60 head3 = n;  
61 }  
62 else{  
63 while(temp.next!=null){  
64 temp = temp.next;  
65 }  
66 temp.next = n;  
67 }  
68 }  
69 public static void shuffleMerge(int size){  
70 Node temp1 = head1;  
71 Node temp2 = head2;  
72 int[] temp = new int[size\*2];  
73 int i =0;  
74 while(temp1 != null && temp2 != null){  
75 temp[i] = temp1.data;  
76 temp[i+1] = temp2.data;  
77 i = i+2;  
78 temp1 = temp1.next;  
79 temp2 = temp2.next;  
80 }  
81 for(i = 0; i<size\*2; i++){  
82 node\_end(temp[i]);  
83 }  
84 }  
85 public static void print\_nodes(){  
86 Node temp = new Node();  
87 temp = head3;  
88 if(temp == null){  
89 System.out.print("\nNo Elements found!\n");  
90 }  
91 else{  
92 while(temp.next!=null){  
93 System.out.print(temp.data+"--->");  
94 temp = temp.next;  
95 }  
96 System.out.print(temp.data+"\n");  
97 }  
98 }  
99 }

**QUESTION 8**

1 import java.util.\*;  
 2 class Node\_Dou{  
 3 int data;  
 4 Node\_Dou next ;  
 5 Node\_Dou pre ;  
 6 Node\_Dou(){  
 7 data = 0;  
 8 next = null;  
 9 pre = null;  
 10 }  
 11 }  
 12 public class Linked\_SetB\_Q8{  
 13 static Node\_Dou head;  
 14 static Node\_Dou tail;  
 15 public static void main(String args[]){  
 16 Scanner sc = new Scanner(System.in);  
 17 int choice = 0;  
 18 do{  
 19 System.out.println(" Enter 1 to add element in front.\n Enter 2 to add elements in back.\n Enter 3 to add in between by index number.\n Enter 4 to add in between with searching element.\n Enter 5 to delete an element from start.\n Enter 6 to delete an element from last.\n Enter 7 to delete an element at specific location.\n Enter 8 to print list in forward direction.\n Enter 9 to print list in Reverse direction.\n Enter 10 to exit.");  
 20 choice = sc.nextInt();  
 21 if(choice==1){  
 22 System.out.print("Enter value: ");  
 23 int a = sc.nextInt();  
 24 front(a);  
 25 }  
 26 else if(choice==2){  
 27 System.out.print("Enter value: ");  
 28 int a = sc.nextInt();  
 29 back(a);  
 30 }  
 31 else if(choice==3){  
 32 System.out.print("Enter index value: ");  
 33 int i = sc.nextInt();  
 34 System.out.print("Enter value: ");  
 35 int a = sc.nextInt();  
 36 midindex(i,a);  
 37 }  
 38 else if(choice==4){  
 39 System.out.print("Enter element after which you want to add an element: ");  
 40 int i = sc.nextInt();  
 41 System.out.print("Enter value: ");  
 42 int a = sc.nextInt();  
 43 midvalue(i,a);  
 44 }  
 45 else if(choice==10) System.out.println("Program terminated.");  
 46 else if(choice==8){  
 47 printForward();  
 48 }  
 49 else if(choice==5){  
 50 delStart();  
 51 }  
 52 else if(choice==9){  
 53 printReverse();  
 54 }  
 55 else if(choice==6){  
 56 delLast();  
 57 }  
 58 else if(choice==7){  
 59 System.out.print("Enter index of value you want to delete: ");  
 60 int a = sc.nextInt();  
 61 delSpecific(a);  
 62 }  
 63 else System.out.println("Invalid Input, Enter again ");  
 64 }while(choice!=10);  
 65 }  
 66 public static void back(int a){  
 67 Node\_Dou n = new Node\_Dou();  
 68 n.data = a;  
 69 if(head == null){  
 70 head = n;  
 71 tail = n;  
 72 }  
 73 else{  
 74 tail.next = n;  
 75 n.pre= tail;  
 76 tail = tail.next;  
 77 }  
 78 }  
 79 public static void printForward(){  
 80 Node\_Dou temp = new Node\_Dou();  
 81 temp = head;  
 82 if(head == null) System.out.println("List is empty.");  
 83 else{  
 84 while(temp!=tail){  
 85 System.out.print(temp.data+" ");  
 86 temp = temp.next;  
 87 }  
 88 System.out.print(tail.data);  
 89 System.out.println();  
 90 }  
 91 }  
 92 public static void printReverse(){  
 93 Node\_Dou temp = new Node\_Dou();  
 94 temp = tail;  
 95 if(head == null) System.out.println("List is empty.");  
 96 else{  
 97 while(temp!=head){  
 98 System.out.print(temp.data+" ");  
 99 temp = temp.pre;  
100 }  
101 System.out.print(head.data);  
102 System.out.println();  
103 }  
104 }  
105 public static void front(int a){  
106 Node\_Dou n = new Node\_Dou();  
107 n.data = a;  
108 n.next=head;  
109 n.pre =null;  
110 head.pre = n;  
111 head = n;  
112 if(head.next==null) tail = n;  
113 }  
114 public static void midindex(int j, int a){  
115 Node\_Dou n = new Node\_Dou();  
116 n.data = a;  
117 Node\_Dou temp = new Node\_Dou();  
118 temp = head;  
119 int r = 0;  
120 if((head == null && j>1) || j<1){  
121 System.out.println("Index out of bound.");  
122 r=1;  
123 }  
124 else if(j==1){  
125 n.next=head;  
126 n.pre = null;  
127 if(head!=null) head.pre=n;  
128 head = n;  
129 if(head.next==null) tail = n;  
130 r=1;  
131 }  
132 else if(j==2){  
133 n.next=head.next;  
134 n.pre = head;  
135 (head.next).pre = n;  
136 head.next = n;  
137 if(n.next==null) tail=n;  
138 r=1;  
139 }  
140 else{  
141 for(int i=0; i<j-2;i++){  
142 if(temp.next==null){  
143 System.out.println("Entered index is out of bound.");  
144 r =1;  
145 break;  
146 }  
147 else temp =temp.next;  
148 }  
149 }  
150 if(r==0){  
151 n.next = temp.next;  
152 n.pre = temp;  
153 if(temp!= tail) (temp.next).pre =n;  
154 temp.next=n;  
155 if(temp == tail) tail = n;  
156 }  
157 }  
158 public static void midvalue(int i, int a){  
159 Node\_Dou n = new Node\_Dou();  
160 n.data = a;  
161 Node\_Dou temp = new Node\_Dou();  
162 temp = head;  
163 int r = 0;  
164 if(head==null){  
165 System.out.println("List is empty.");  
166 r=1;  
167 }  
168 else{  
169 while(temp.data!=i ){  
170 if(temp.next==null){  
171 r=1;  
172 System.out.println("Entered number not found in list.");  
173 break;  
174 }  
175 temp = temp.next;  
176 }  
177 }   
178 if(r==0){  
179 n.next = temp.next;  
180 n.pre = temp;  
181 if(temp!= tail) (temp.next).pre =n;  
182 temp.next=n;  
183 if(temp == tail) tail = n;  
184 }  
185 }  
186 public static void delStart(){  
187 if(head==null) System.out.println(" List is empty.");  
188 else if(head==tail){  
189 head = null;  
190 tail = null;  
191 }  
192 else{  
193 head = head.next;  
194 head.pre = null;  
195 }  
196 }  
197 public static void delLast(){  
198 Node\_Dou temp = new Node\_Dou();  
199 temp = head;  
200 if(head==null) System.out.println(" List is empty.");  
201 else if(head==tail){  
202 head = null;  
203 tail = null;  
204 }  
205 else{  
206 tail = tail.pre;  
207 tail.next = null;  
208 }  
209 }  
210 public static void delSpecific(int j){  
211 Node\_Dou temp = new Node\_Dou();  
212 temp = head;  
213 if(head==null) System.out.println(" List is empty.");  
214 else if(j ==1){  
215 if(head==tail) tail = null;  
216 head = head.next;  
217 }  
218 else if(j==2 && head == tail){  
219 System.out.println("Entered index is out of bound.");  
220 }   
221 else if (j>1){  
222 int r = 0;  
223 for(int i =0; i<j-2;i++){  
224 if(temp.next==null || (temp.next.next==null)){  
225 System.out.println("Entered index is out of bound.");  
226 r=1;  
227 break;  
228 }  
229 else{  
230 temp=temp.next;  
231 }  
232 }  
233 if (r==0){  
234 if(temp.next == tail) tail = temp;  
235 else temp.next.next.pre = temp;  
236 temp.next= (temp.next).next;  
237 }  
238 }  
239 }  
240 }  
241

**QUESTION 9**

1 import java.util.\*;  
 2 public class Linked\_SetB\_Q9{  
 3 static Node head,head1;  
 4 public static void main(String args[]){  
 5 Scanner sc=new Scanner(System.in);  
 6 int size,element;  
 7 System.out.println("Enter the number of elements of list: ");  
 8 size=sc.nextInt();  
 9 System.out.println("Elements of list: ");  
10 for(int i=0;i<size;i++){  
11 System.out.println("Enter the "+(i+1)+" position of element of list: ");  
12 element=sc.nextInt();  
13 last\_node(element);  
14 }  
15 System.out.println("Enter the position from which to rotaate: ");  
16 int pos=sc.nextInt();  
17 rotate(pos,size);  
18 print\_nodes();  
19 }  
20 public static void rotate(int rot,int size){  
21 Node temp = new Node();  
22 temp = head;  
23 int x[]=new int[size];  
24 for(int i=0;i<rot;i++){  
25 temp = temp.next;  
26 }  
27 int i=0;  
28 while(temp!=null){   
29 x[i]=temp.data;  
30 i=i+1;  
31 temp=temp.next;  
32   
33 }  
34 temp=head;  
35 for(int j = 0; j<rot; j++){  
36 x[i]=temp.data;  
37 i=i+1;  
38 temp = temp.next;  
39 }  
40 for(int k = 0; k<size; k++){  
41 last\_node(x[k],2);  
42 }  
43 }  
44   
45   
46 public static void print\_nodes(){  
47 Node temp = new Node();  
48 temp = head1;  
49 if(temp == null){  
50 System.out.print("\nNo Elements found!\n");  
51 }  
52 else{  
53 while(temp.next!=null){  
54 System.out.print(temp.data+"--->");  
55 temp = temp.next;  
56 }  
57 System.out.print(temp.data+"\n");  
58 }  
59 }  
60 public static void last\_node(int value){  
61 Node new\_node = new Node();  
62 new\_node.data = value;  
63 Node temp = new Node();  
64 temp = head;  
65 if(head==null){  
66 head = new\_node;  
67 }  
68 else{  
69 while(temp.next!=null){  
70 temp = temp.next;  
71 }  
72 temp.next = new\_node;   
73 }   
74 }  
75   
76   
77 public static void last\_node(int value,int r){  
78 Node new\_node = new Node();  
79 new\_node.data = value;  
80 Node temp = new Node();  
81 temp = head1;  
82 if(head1==null){  
83 head1 = new\_node;  
84 }  
85 else{  
86 while(temp.next!=null){  
87 temp = temp.next;  
88 }  
89 temp.next = new\_node;   
90 }   
91 }  
92   
93 }

**QUESTION 10**

1 import java.util.\*;  
 2 class node{  
 3 int data;  
 4 node next ;  
 5 node pre ;  
 6 node(){  
 7 data = 0;  
 8 next = null;  
 9 pre = null;  
 10 }  
 11 }  
 12 public class LinkedListSecB10{  
 13 static node head;  
 14 public static void main(String args[]){  
 15 Scanner sc = new Scanner(System.in);  
 16 System.out.print("Enter the size of list: ");  
 17 int size = sc.nextInt();  
 18 for(int i =0; i<size; i++){  
 19 System.out.print("Enter "+(i+1)+" value: ");  
 20 int a = sc.nextInt();  
 21 node\_end(a);  
 22 }  
 23 int choice = 0;  
 24 do{  
 25 System.out.println("\n Enter 1 to add in between by index number.\n Enter 2 to delete an element at specific location.\n Enter 3 to print list.\n Enter 4 to exit.");  
 26 choice = sc.nextInt();  
 27 if(choice==1){  
 28 System.out.print("Enter index value: ");  
 29 int i = sc.nextInt();  
 30 System.out.print("Enter value: ");  
 31 int a = sc.nextInt();  
 32 add\_node\_index(i,a);  
 33 }  
 34   
 35 else if(choice==4){  
 36 System.out.println("Program terminated.");  
 37 }  
 38 else if(choice==3){  
 39 print\_nodes();  
 40 }  
 41 else if(choice==2){  
 42 System.out.print("Enter index of value you want to delete: ");  
 43 int a = sc.nextInt();  
 44 del\_Index(a);  
 45 }  
 46 else System.out.println("Invalid Input, Enter again ");  
 47 }while(choice!=4);  
 48 }  
 49 public static void node\_end(int a){  
 50 node n = new node();  
 51 n.data = a;  
 52 node temp = new node();  
 53 temp = head;  
 54 if(head == null){  
 55 head = n;  
 56 head.next = head;  
 57 }  
 58 else{  
 59 while(temp.next!=head){  
 60 temp = temp.next;  
 61 }  
 62 n.next = head;  
 63 temp.next = n;  
 64 }  
 65 }  
 66 public static void print\_nodes(){  
 67 node temp = new node();  
 68 temp = head;  
 69 if(temp == null){  
 70 System.out.print("\nNo Elements found!\n");  
 71 }  
 72 else{  
 73 while(temp.next!=null){  
 74 System.out.print(temp.data+"--->");  
 75 temp = temp.next;  
 76 }  
 77 System.out.print(temp.data+"\n");  
 78 }  
 79 }  
 80 public static void add\_node\_index(int index,int data\_){   
 81 node temp = new node();  
 82 node new\_node = new node();  
 83 new\_node.data = data\_;  
 84 int ctr = 0;  
 85 temp = head;  
 86 if(index == 0 ){  
 87 new\_node.next = head;  
 88 head = new\_node;  
 89 ctr = 1;  
 90 }  
 91 else if(index == 1){  
 92 new\_node.next = head.next;  
 93 head.next = new\_node;  
 94 ctr = 1;  
 95 }  
 96   
 97 else {  
 98 for(int i = 0 ; i<index-1 ; i++){  
 99 if(temp.next==null){  
100 System.out.println("Entered index is out of bound.");  
101 ctr = 1;  
102 break;  
103 }  
104 else  
105 temp =temp.next;  
106 }  
107 if(ctr==0){  
108 new\_node.next = temp.next;  
109 temp.next=new\_node;  
110 }  
111 }  
112   
113   
114 }  
115   
116 public static void del\_Index(int j){  
117 node temp = new node();  
118 temp = head;  
119 if(head==null) System.out.println(" List is empty.");  
120 else if(j ==1){  
121 while(temp.next!=head){  
122 System.out.print(temp.data+" ");  
123 temp = temp.next;  
124 }  
125 head = head.next;  
126 temp.next = head;  
127 }  
128 else if (j>0){  
129 int r = 0;  
130 for(int i =0; i<j-2;i++){  
131 if(temp.next==head){  
132 System.out.println("Entered index is out of bound.");  
133 r=1;  
134 break;  
135 }  
136 else{  
137 temp=temp.next;  
138 }  
139 }  
140 if (r==0) temp.next= (temp.next).next;  
141 }  
142 }  
143 }

**SET C**

**QUESTION 11**

1 import java.util.\*;  
 2 public class Linked\_SetC\_Q11{  
 3 static Node head1,head2,head;  
 4 public static void main(String args[]){  
 5 Scanner sc=new Scanner(System.in);  
 6 int size,element;  
 7 System.out.println("Enter the number of elements of list: ");  
 8 size=sc.nextInt();  
 9 int x[]=new int[size];  
10 System.out.println("Elements of list1: ");  
11 for(int i=0;i<size;i++){  
12 System.out.println("Enter the "+(i+1)+" position of element of list1: ");  
13 element=sc.nextInt();  
14 x[i]=element;  
15 }  
16 min\_max(size,x);  
17 System.out.print("\nThe Sorted Nodes are as follows: ");  
18 print\_nodes();  
19 }  
20 public static void min\_max(int size,int[] x){  
21 ArrayList<Integer> l = new ArrayList<Integer>();  
22 Arrays.sort(x);  
23 for(int p=0;p<size/2;p=p+1){  
24 l.add(x[p]);  
25 l.add(x[size-p-1]);  
26 }  
27 if(size%2!=0){  
28 l.add(x[size/2]);  
29 }  
30 for(int k = 0; k<size; k++){  
31 last\_node(l.get(k));  
32   
33 }  
34 }  
35 public static void print\_nodes(){  
36 Node temp = new Node();  
37 temp = head;  
38 if(temp == null){  
39 System.out.print("\nNo Elements found!\n");  
40 }  
41 else{  
42 while(temp.next!=null){  
43 System.out.print(temp.data+"--->");  
44 temp = temp.next;  
45 }  
46 System.out.print(temp.data+"\n");  
47 }  
48 }  
49 public static void last\_node(int value){  
50 Node new\_node = new Node();  
51 new\_node.data = value;  
52 Node temp = new Node();  
53 temp = head;  
54 if(head==null){  
55 head = new\_node;  
56 }  
57 else{  
58 while(temp.next!=null){  
59 temp = temp.next;  
60 }  
61 temp.next = new\_node;   
62 }   
63 }  
64   
65   
66   
67 }

**QUESTION 12**

1 import java.util.\*;  
 2 public class Linked\_SetC\_Q12{  
 3 static Node head;  
 4 public static void main(String args[]){  
 5 Scanner sc = new Scanner(System.in);  
 6 System.out.print("Enter the number of students: ");  
 7 int size = sc.nextInt();  
 8 for(int i =0; i<size; i++){  
 9 Node\_end(i+1);  
10 }  
11 System.out.print("Enter the value at which the student should get eliminated : ");  
12 int value = sc.nextInt();  
13 eliminate(value);  
14 }  
15 public static void Node\_end(int a){  
16 Node n = new Node();  
17 n.data = a;  
18 Node temp = new Node();  
19 temp = head;  
20 if(head == null){  
21 head = n;  
22 head.next = head;  
23 }  
24 else{  
25 while(temp.next!=head){  
26 temp = temp.next;  
27 }  
28 n.next = head;  
29 temp.next = n;  
30 }  
31 }  
32 public static void eliminate(int value){  
33 int i = 0;  
34 while(i<1){  
35 if(head.next == head){  
36 System.out.println(head.data+" is the winner.");  
37 i = 2;  
38 }  
39 else{  
40 del\_position(value);  
41 }  
42 }  
43 }  
44 public static void del\_position(int j){  
45 Node temp = new Node();  
46 temp = head;  
47 if(j>0){  
48 int r = 0;  
49 for(int i =0; i<j-2;i++){  
50 temp=temp.next;  
51 }  
52 if (r==0){  
53 temp.next= (temp.next).next;  
54 head = temp.next;  
55 }  
56 }  
57 }  
58 }

**BINAY 19CSU370**

**IOT-A**