

ASSIGNMENT – 3

2022503303

QUESTION 1:

LINKED LIST

```
Settings linkedlist3303.java x J LinkedLis
J linkedlist3303.java > linkedlist3303 > main(String[])
1  import java.util.Scanner;
2
3  class Node{
4      int data;
5      Node next;
6      Node(int data){
7          this.data = data;
8          this.next = null;
9      }
10 };
11 public class linkedlist3303{
12     Node head;
13     linkedlist3303(){
14         head = null;
15     }
16     public void insert(int data){
17         if(head == null) {insertatbeg(data);return;}
18         Node newnode = new Node(data);
19         Node temp = head;
20         while(temp.next != null){
21             temp = temp.next;
22         }
23         temp.next = newnode;
24         newnode.next = null;
25     }
26     public void insertatbeg(int data){
27         Node newnode = new Node(data);
28         newnode.next = head;
29         head = newnode;
30     }
31     public void insertatpos(int pos, int data){
32         Node newnode = new Node(data);
33         Node temp = head;
34         for(int i=0;i<pos-2;i++){
35             temp = temp.next;
36         }
37         newnode.next = temp.next;
38         temp.next = newnode;
39     }
40     public void insertatend(int data){
41         Node newnode = new Node(data);
42         Node temp = head;
43         while(temp.next != null){
44             temp = temp.next;
45         }
46         temp.next = newnode;
47         newnode.next = null;
48     }
49     public int insertafter(int data, int ser){
50         Node newnode = new Node(data);
51         Node temp = head;
52         while(temp.data != ser && temp != null){
53             temp = temp.next;
54         }
55         if(temp == null) return -1;
56         newnode.next = temp.next;
57         temp.next = newnode;
58         return 0;
59     }
60     public int insertbefore(int data, int ser){
```

```

61  ✓      if(head.data == ser){
62  ●      insertatbeg(data);
63          return 1;
64      }
65      Node newnode = new Node(data);
66      Node temp = head;
67  ✓      while(temp.next.data != ser && temp != null){
68          temp = temp.next;
69      }
70      if(temp == null) return -1;
71      newnode.next = temp.next;
72      temp.next = newnode;
73      return 0;
74  }
75  ✓      public void deletebyval(int data){
76      Node temp = head;
77  ✓      if(head.data == data){
78          head = head.next;
79      }
80  ✓      else{
81  ✓          while(temp.next.data != data){
82              temp = temp.next;
83          }
84          temp.next = temp.next.next;
85      }
86  }
87  ✓      public void deletebypos(int pos){
88  ✓      if(pos == 0){
89          head = head.next;

```

```

90          return;
91      }
92      Node temp = head;
93      for(int i=0;i<pos-2;i++){
94          temp = temp.next;
95      }
96      temp = temp.next;
97  }
98      public int search(int data){
99      Node temp = head;
100      int i=0;
101      while(temp != null){
102          if(temp.data == data) return i;
103          i++;
104      }
105      return -1;
106  }
107      public void duplicate(){
108          sort(count:0);
109          Node tm = head;
110          Node tm2 = head.next;
111          while(tm2 != null){
112              if(tm.data != tm2.data){
113                  tm = tm.next;
114                  tm.data = tm2.data;
115              }
116              tm2 = tm2.next;
117          }
118          tm.next = null;

```

```

119     }
120     public void display(){
121         Node temp = head;
122         if(head == null) System.out.println(x:"null");
123         else{
124             while(temp != null){
125                 System.out.println(temp.data + " -> ");
126                 temp = temp.next;
127             }
128             System.out.print(s:"null");
129             System.out.println();
130         }
131     }
132     public void display(Node hd){
133         Node temp = hd;
134         if(head == null) System.out.println(x:"null");
135         else{
136             while(temp != null){
137                 System.out.print(temp.data + " -> ");
138                 temp = temp.next;
139             }
140             System.out.print(s:"null");
141             System.out.println();
142         }
143     }
144     public int countoccur(int ele){
145         Node temp = head;
146         int c = 0;
147         while(temp.next != null){

```

```

148             if(temp.data == ele) c++;
149             temp = temp.next;
150         }
151         return c;
152     }
153     public void reverse(){
154         Node temp = head;
155         Node prev = null;
156         while(temp != null){
157             Node front = temp.next;
158             temp.next = prev;
159             prev = temp;
160             temp = front;
161         }
162         head = prev;
163         display();
164     }
165     public void sort(int count){
166         Node t1 = head;
167         while(t1 != null){
168             Node t2 = t1.next;
169             while(t2 != null){
170                 if(t1.data > t2.data){
171                     int temp = t2.data;
172                     t2.data = t1.data;
173                     t1.data = temp;
174                 }
175                 t2 = t2.next;
176             }

```

```

177         t1 = t1.next;
178     }
179 }
180 public Node concat(linkedlist3303 ls, linkedlist3303 l2){
181     if(ls.head == null) ls.head = l2.head;
182     else{
183         Node temp1 = (Node) ls.head;
184         while(temp1.next != null){
185             temp1 = temp1.next;
186         }
187         temp1.next = (Node) l2.head;
188     }
189     return (Node) ls.head;
190 }
191 Run | Debug
192 public static void main(String[] args){
193     linkedlist3303 ls = new linkedlist3303();
194     try (Scanner sc = new Scanner(System.in)) {
195         int count = 0;
196         System.out.println(x:" Linked List ");
197         while(true){
198             System.out.println(x:" Enter Options : ");
199             System.out.println(x:"1. Insertion \n2.Deletion \n3.Search \n4.Reverse \n5.Sort \n6.Count Occurance \n7. Concatenation");
200             int op1 = sc.nextInt();
201             switch(op1){
202                 case 1:
203                     System.out.println(x:" Enter Number To Insert : ");
204                     int num = sc.nextInt();
205                     System.out.println(x:" Options : \n--> Using Position (1)\n--> Using Values(2)");
206                     int op2 = sc.nextInt();
207                     switch(op2){
208                         case 1:
209                             System.out.println(x:" Enter Position To Insert : ");
210                             int pos = sc.nextInt();
211                             if(pos == 0) ls.insertatbeg(num);
212                             else if(pos == count) ls.insertatend(num);
213                             else if(pos > count) System.out.println(x:" INVALID POSITION");
214                             else if(pos > 0) ls.insertatpos(pos, num);
215                             break;
216                         case 2:
217                             System.out.println(x:" Enter Value To Search : ");
218                             int ser = sc.nextInt();
219                             System.out.println(x:" Enter After/Before (0/1): ");
220                             int flag = sc.nextInt();
221                             if(flag == 0) if(ls.insertafter(num, ser) == -1) System.out.println(x:" Value Not Found !");
222                             else if(flag == 1) if(ls.insertbefore(num, ser) == -1) System.out.println(x:" Value Not Found !");
223                             break;
224                     }
225                     count ++;
226                     break;
227                 case 2:
228                     System.out.println(x:" Options : \n--> Using Position (1)\n--> Using Values(2)");
229                     int op3 = sc.nextInt();
230                     if(op3 == 1) {
231                         System.out.println(x:" Enter Position To DEL : ");
232                         int pos = sc.nextInt();
233                         if(pos > count) System.out.println(x:" Invalid Position ");

```

```

233         else{
234             ls.deletebypos(pos);
235         }
236     }
237     else if(op3 == 2){
238         System.out.println(x:" Enter Number To DEL : ");
239         ls.deletebyval(sc.nextInt());
240     }
241     count --;
242     break;
243 case 3:
244     System.out.println(x:" Enter Element to Search : ");
245     int ind = ls.search(sc.nextInt());
246     if(ind != -1) System.out.println("Element found at "+ind+" position !");
247     else System.out.println(x:" Element Not Found !");
248     break;
249 case 4:
250     ls.reverse();
251     break;
252 case 5:
253     ls.sort(count);
254     break;
255 case 6:
256     System.out.println(x:" Enter Element to Count : ");
257     int ele = sc.nextInt();
258     int occur = ls.countoccur(ele);
259     System.out.println("Number of "+ele+"'s : "+ occur);
260     break;
261 case 7:
262     System.out.println(x:" Enter the number of elements to insert : ");
263     int size = sc.nextInt();
264     linkedlist3303 l2 = new linkedlist3303();
265     while(size > 0){
266         System.out.println(x:" Enter the Elements : ");
267         l2.insert(sc.nextInt());
268         size--;
269     }
270     System.out.println(x:" LINKED LIST 1 : ");
271     ls.display();
272     System.out.println(x:" LINKED LIST 2 : ");
273     l2.display();
274     Node hd = ls.concat(ls, l2);
275     System.out.println(x:" CONCATENATED : ");
276     ls.display(hd);
277     break;
278 }
279 System.out.println(x:" Do You Want to Display ?");
280 int flag1 = sc.nextInt();
281 if(flag1 == 1) ls.display();
282 System.out.println(x:" Do You Want to Continue ?");
283 int flag2 = sc.nextInt();
284 if(flag2 != 1) break;
285 }
286 }
287 };
288 };

```

OUTPUT:

Insertion

```
** Linked List **
[+] Enter Options :
1. Insertion
2.Deletion
3.Search
4.Reverse
5.Sort
6.Count Occurance
7. Concatenation
1
[-] Enter Number To Insert :
44
[-] Options :
--> Using Position (1)
--> Using Values(2)
1
[-] Enter Position To Insert :
0
[+] Do You Want to Display ?
1
44 -> null
[+] Do You Want to Continue ?
1
[+] Enter Options :
1. Insertion
2.Deletion
3.Search
4.Reverse
5.Sort
6.Count Occurance
7. Concatenation
1
[-] Enter Number To Insert :
66
[-] Options :
--> Using Position (1)
--> Using Values(2)
2
[-] Enter Value To Search :
44
[-] Enter After/Before (0/1):
0
[+] Do You Want to Display ?
1
44 -> 66 -> null
```

Deletion

```
[+] Enter Options :
1. Insertion
2.Deletion
3.Search
4.Reverse
5.Sort
6.Count Occurance
7. Concatenation
2
[-] Options :
--> Using Position (1)
--> Using Values(2)
1
[-] Enter Position To DEL :
0
[+] Do You Want to Display ?
1
66 -> null
[+] Do You Want to Continue ?
1
[+] Enter Options :
1. Insertion
2.Deletion
3.Search
4.Reverse
5.Sort
6.Count Occurance
7. Concatenation
2
[-] Options :
--> Using Position (1)
--> Using Values(2)
2
[-] Enter Number To DEL :
66
[+] Do You Want to Display ?
1
null
```

Search

```
44 -> 77 -> null
[+] Do You Want to Continue ?
1
[+] Enter Options :
1. Insertion
2.Deletion
3.Search
4.Reverse
5.Sort
6.Count Occurance
7. Concatenation
3
[-] Enter Element to Search :
44
[-] Element found at 0 position !
```

Reverse

```
44 -> 77 -> null
[+] Do You Want to Continue ?
1
[+] Enter Options :
1. Insertion
2.Deletion
3.Search
4.Reverse
5.Sort
6.Count Occurance
7. Concatenation
4
77 -> 44 -> null
```

Sort

```
77 -> 44 -> null
[+] Do You Want to Continue ?
1
[+] Enter Options :
1. Insertion
2.Deletion
3.Search
4.Reverse
5.Sort
6.Count Occurance
7. Concatenation
5
[+] Do You Want to Display ?
1
44 -> 77 -> null
```


Count Occurance

```
[+] Enter Options :
1. Insertion
2.Deletion
3.Search
4.Reverse
5.Sort
6.Count Occurance
7. Concatenation
6
[+] Enter Element to Count :
44
Number of 44's : 1
```

Concatenated

```
44 -> 77 -> null
[+] Do You Want to Continue ?
1
[+] Enter Options :
1. Insertion
2.Deletion
3.Search
4.Reverse
5.Sort
6.Count Occurance
7. Concatenation
7
[+] Enter the number of elements to insert :
2
[-] Enter the Elements :
88
[-] Enter the Elements :
99
[-] LINKED LIST 1 :
44 -> 77 -> null
[-] LINKED LIST 2 :
88 -> 99 -> null
[*] CONCATENATED :
44 -> 77 -> 88 -> 99 -> null
```

Duplicate

```
11 -> 22 -> 11 -> 22 -> 33 -> null
[+] Do You Want to Display ?
0
[+] Do You Want to Continue ?
1
[+] Enter Options :
1.Insertion
2.Deletion
3.Search
4.Reverse
5.Sort
6.Count Occurance
7.Concatenation
8.Duplicate
8
11 -> 22 -> 33 -> null
```

QUESTION 2:

STACK

```
J Stack3303.java > Stack3303 > display()
1  import java.util.Scanner;
2  class Node{
3      int data;
4      Node next;
5      Node(int data){
6          this.data = data;
7          this.next = null;
8      }
9  }
10 public class Stack3303 {
11     Node head;
12     Stack3303(){
13         head = null;
14     }
15     public void push(int data){
16         Node newnode = new Node(data);
17         if(head == null){
18             newnode.next = null;
19             head = newnode;
20         }
21         else{
22             newnode.next = head;
23             head = newnode;
24         }
25     }
26     public void pop(){
27         if(head == null) System.out.println(x: "[+] Stack is Empty !");
28         else
29             head = head.next;
30     }
31     public void top(){
32         if(head == null) System.out.println(x: "[+] Stack is Empty !");
33         else
34             System.out.println(head.data);
35     }
36     public void display(){
37         Node tp = head;
38         System.out.println(x: "[+] Elements : ");
39         while(tp != null){
40             System.out.print(tp.data + " -> ");
41             tp = tp.next;
42         }
43         System.out.println(x: "null");
44     }
45     public static void main(String[] args){
46         Scanner sc = new Scanner(System.in);
47         Stack3303 st = new Stack3303();
48         System.out.println(x: "** STACK **");
49         while(true){
50             System.out.println(x: "[+] Operations : \n1.PUSH \n2.POP \n3.TOP \n4.DISPLAY \n5.EXIT");
51             int op = sc.nextInt();
52             switch (op) {
53                 case 1:
54                     System.out.println(x: "[-] Enter Element to Push");
55                     st.push(sc.nextInt());
56                     break;
57                 case 2:
58                     st.pop();
59                     break;
60                 case 3:
61                     st.top();
62                     break;
63                 case 4:
64                     st.display();
65                     break;
66                 case 5:
67                     break;
68             }
69         }
70     }
71 }
```

OUTPUT:

```
[+] Operations :
1.PUSH
2.POP
3.TOP
4.DISPLAY
5.EXIT
4.DISPLAY
4.DISPLAY
5.EXIT
1
[-] Enter Element to Push
22
[+] Operations :
1.PUSH
2.POP
3.TOP
4.DISPLAY
5.EXIT
1
[-] Enter Element to Push
4
[+] Operations :
1.PUSH
2.POP
3.TOP
4.DISPLAY
5.EXIT
4
[+] Elements :
4 -> 22 -> null
```

[+] Operations :

- 1.PUSH
- 2.POP
- 3.TOP
- 4.DISPLAY
- 5.EXIT

2

[+] Operations :

- 1.PUSH
- 2.POP
- 3.TOP
- 4.DISPLAY
- 5.EXIT

4

[+] Elements :

22 -> null

[+] Operations :

- 1.PUSH
- 2.POP
- 3.TOP
- 4.DISPLAY
- 5.EXIT

1

[-] Enter Element to Push

66

[+] Operations :

- 1.PUSH
- 2.POP
- 3.TOP
- 4.DISPLAY

5.EXIT

4

[+] Elements :

66 -> 22 -> null

[+] Operations :

- 1.PUSH
- 2.POP
- 3.TOP
- 4.DISPLAY
- 5.EXIT

QUESTION 3 (ASSIGNEMENT – 2)Q4

MAGIC SQUARE:

```
J Magic3303.java > ...
1  import java.util.Scanner;
2
3  public class Magic3303 {
4      int[][] arr = new int[3][3];
5      public void getinput(){
6          Scanner sc = new Scanner(System.in);
7          int mid = sc.nextInt();
8          arr[0][0] = mid + 1;
9          arr[0][1] = mid - 4;
10         arr[0][2] = mid + 3;
11         arr[1][0] = mid + 2;
12         arr[1][1] = mid;
13         arr[1][2] = mid - 2;
14         arr[2][0] = mid - 3;
15         arr[2][1] = mid + 4;
16         arr[2][2] = mid - 1;
17     }
18     public void display(){
19         for(int i=0;i<3;i++){
20             for(int j=0;j<3;j++){
21                 System.out.printf(format:"%3d",arr[i][j]);
22             }
23             System.out.println();
24         }
25     }
26 }
Run | Debug
27 public static void main(String[] args){
28     Magic3303 magic = new Magic3303();
29     magic.getinput();
30     magic.display();
31 }
32 }
```

OUTPUT:

```
PS D:\java> d:; cd 'd:\java'; & 'C:\Program Files\Java\jdk-17\bin\java.exe' '-XX:+Show
oaming\Code\User\workspaceStorage\0879bcb9d0085e0215c39822a6b2224b\redhat.java\jdt_ws\j
15
16 11 18
17 15 13
12 19 14
PS D:\java> 
```