

S. No.	Date	Title	Page No.	Teacher's Sign / Remarks
1	21/12/23	Swapping two variables using dynamic memory		
2	21/12/23	Stack Implementation	11/10	
2	11/01/24	Infix to Postfix expression	11/10	
3	11/01/24	Queue Implementation Circular Queue Implementation	11/11	
4	11/01/24	Singly linked list Implementation	11/11	
5	18/01/24	linked list Operations deallocate-MinStack	10/181	
6	25/01/24	linked list: Sort, Reverse, Concat linked list - Stack and Queue operations	10/186	
7	21/02/24	Doubly linked list deallocate-Reversing linked list	6/101	
8	15/02/24	Construct binary search tree deallocate-Split Linked list in parts	10/168	
9	22/02/24	Traverse using BFS graph is connected-DFS deallocate-Rotate list	5/108	
10	29/02/24	Hashing HackerRank-Split		

Practice Q
 Output:
 1) enter 1 -
 3 - Jo
 enter M
 enter F
 enter 1
 3 do
 3
 enter
 enter
 3 -
 5
 cu
 a
 2) Er
 en
 ca
 c
 e
 6

Practice Questions

Output 1:

- 1) enter 1 - to create a account 2 - to withdraw
3 - to deposit 4 - to check balance 5 - exit
enter the name - Reshma. V
enter the age - 19
enter 1 - to create a account 2 - to withdraw
3 - to deposit 4 - to check balance 5 - exit
3
enter the amount to deposit : 10000
enter 1 - to create a account 2 - to withdraw
3 - to deposit 4 - to check balance 5 - exit
3
enter the amount to deposit : 10000
enter 1 - to create a account 2 - to withdraw
3 - to deposit 4 - to check balance 5 - exit
5

Output 2:

- 2) Enter the number of strings : 4
enter strings : Hello
enter string 2 : Bye
enter string 3 : world
enter string 4 : Hi
display strings

Bye

Hello

Hi

world

Output 3:

- 3) read n : 2
read m : 2

read the elements :

- 3
4 enter the key element : 2
element found
Output 4
4) cover a larger string; useful
enter a substring to search for: use
searching found
Output 9
5) Read n : 5
Over the elements of the array
1
2
3
4
5 enter the number to find the first occurrence : 2
first occurrence of 2 is at 3
Output 6
6) Read n : 5
Read the elements
1
2
3
4
5
6
7
8
9
10 enter the key element : 8
element found at 3

Outflow
7) even
scaler
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
101
102
103
104
105
106
107
108
109
110
111
112
113
114
115
116
117
118
119
120
121
122
123
124
125
126
127
128
129
130
131
132
133
134
135
136
137
138
139
140
141
142
143
144
145
146
147
148
149
150
151
152
153
154
155
156
157
158
159
160
161
162
163
164
165
166
167
168
169
170
171
172
173
174
175
176
177
178
179
180
181
182
183
184
185
186
187
188
189
190
191
192
193
194
195
196
197
198
199
200
201
202
203
204
205
206
207
208
209
210
211
212
213
214
215
216
217
218
219
220
221
222
223
224
225
226
227
228
229
230
231
232
233
234
235
236
237
238
239
240
241
242
243
244
245
246
247
248
249
250
251
252
253
254
255
256
257
258
259
260
261
262
263
264
265
266
267
268
269
270
271
272
273
274
275
276
277
278
279
280
281
282
283
284
285
286
287
288
289
290
291
292
293
294
295
296
297
298
299
300
301
302
303
304
305
306
307
308
309
310
311
312
313
314
315
316
317
318
319
320
321
322
323
324
325
326
327
328
329
330
331
332
333
334
335
336
337
338
339
340
341
342
343
344
345
346
347
348
349
350
351
352
353
354
355
356
357
358
359
360
361
362
363
364
365
366
367
368
369
370
371
372
373
374
375
376
377
378
379
380
381
382
383
384
385
386
387
388
389
390
391
392
393
394
395
396
397
398
399
400
401
402
403
404
405
406
407
408
409
410
411
412
413
414
415
416
417
418
419
420
421
422
423
424
425
426
427
428
429
430
431
432
433
434
435
436
437
438
439
440
441
442
443
444
445
446
447
448
449
450
451
452
453
454
455
456
457
458
459
460
461
462
463
464
465
466
467
468
469
470
471
472
473
474
475
476
477
478
479
480
481
482
483
484
485
486
487
488
489
490
491
492
493
494
495
496
497
498
499
500
501
502
503
504
505
506
507
508
509
510
511
512
513
514
515
516
517
518
519
520
521
522
523
524
525
526
527
528
529
530
531
532
533
534
535
536
537
538
539
540
541
542
543
544
545
546
547
548
549
550
551
552
553
554
555
556
557
558
559
550
551
552
553
554
555
556
557
558
559
560
561
562
563
564
565
566
567
568
569
560
561
562
563
564
565
566
567
568
569
570
571
572
573
574
575
576
577
578
579
580
581
582
583
584
585
586
587
588
589
590
591
592
593
594
595
596
597
598
599
600
601
602
603
604
605
606
607
608
609
610
611
612
613
614
615
616
617
618
619
610
611
612
613
614
615
616
617
618
619
620
621
622
623
624
625
626
627
628
629
620
621
622
623
624
625
626
627
628
629
630
631
632
633
634
635
636
637
638
639
630
631
632
633
634
635
636
637
638
639
640
641
642
643
644
645
646
647
648
649
640
641
642
643
644
645
646
647
648
649
650
651
652
653
654
655
656
657
658
659
650
651
652
653
654
655
656
657
658
659
660
661
662
663
664
665
666
667
668
669
660
661
662
663
664
665
666
667
668
669
670
671
672
673
674
675
676
677
678
679
670
671
672
673
674
675
676
677
678
679
680
681
682
683
684
685
686
687
688
689
680
681
682
683
684
685
686
687
688
689
690
691
692
693
694
695
696
697
698
699
690
691
692
693
694
695
696
697
698
699
700
701
702
703
704
705
706
707
708
709
700
701
702
703
704
705
706
707
708
709
710
711
712
713
714
715
716
717
718
719
710
711
712
713
714
715
716
717
718
719
720
721
722
723
724
725
726
727
728
729
720
721
722
723
724
725
726
727
728
729
730
731
732
733
734
735
736
737
738
739
730
731
732
733
734
735
736
737
738
739
740
741
742
743
744
745
746
747
748
749
740
741
742
743
744
745
746
747
748
749
750
751
752
753
754
755
756
757
758
759
750
751
752
753
754
755
756
757
758
759
760
761
762
763
764
765
766
767
768
769
760
761
762
763
764
765
766
767
768
769
770
771
772
773
774
775
776
777
778
779
770
771
772
773
774
775
776
777
778
779
780
781
782
783
784
785
786
787
788
789
780
781
782
783
784
785
786
787
788
789
790
791
792
793
794
795
796
797
798
799
790
791
792
793
794
795
796
797
798
799
800
801
802
803
804
805
806
807
808
809
800
801
802
803
804
805
806
807
808
809
810
811
812
813
814
815
816
817
818
819
810
811
812
813
814
815
816
817
818
819
820
821
822
823
824
825
826
827
828
829
820
821
822
823
824
825
826
827
828
829
830
831
832
833
834
835
836
837
838
839
830
831
832
833
834
835
836
837
838
839
840
841
842
843
844
845
846
847
848
849
840
841
842
843
844
845
846
847
848
849
850
851
852
853
854
855
856
857
858
859
850
851
852
853
854
855
856
857
858
859
860
861
862
863
864
865
866
867
868
869
860
861
862
863
864
865
866
867
868
869
870
871
872
873
874
875
876
877
878
879
870
871
872
873
874
875
876
877
878
879
880
881
882
883
884
885
886
887
888
889
880
881
882
883
884
885
886
887
888
889
890
891
892
893
894
895
896
897
898
899
890
891
892
893
894
895
896
897
898
899
900
901
902
903
904
905
906
907
908
909
900
901
902
903
904
905
906
907
908
909
910
911
912
913
914
915
916
917
918
919
910
911
912
913
914
915
916
917
918
919
920
921
922
923
924
925
926
927
928
929
920
921
922
923
924
925
926
927
928
929
930
931
932
933
934
935
936
937
938
939
930
931
932
933
934
935
936
937
938
939
940
941
942
943
944
945
946
947
948
949
940
941
942
943
944
945
946
947
948
949
950
951
952
953
954
955
956
957
958
959
950
951
952
953
954
955
956
957
958
959
960
961
962
963
964
965
966
967
968
969
960
961
962
963
964
965
966
967
968
969
970
971
972
973
974
975
976
977
978
979
970
971
972
973
974
975
976
977
978
979
980
981
982
983
984
985
986
987
988
989
980
981
982
983
984
985
986
987
988
989
990
991
992
993
994
995
996
997
998
999
990
991
992
993
994
995
996
997
998
999
1000
1001
1002
1003
1004
1005
1006
1007
1008
1009
1000
1001
1002
1003
1004
1005
1006
1007
1008
1009
1010
1011
1012
1013
1014
1015
1016
1017
1018
1019
1010
1011
1012
1013
1014
1015
1016
1017
1018
1019
1020
1021
1022
1023
1024
1025
1026
1027
1028
1029
1020
1021
1022
1023
1024
1025
1026
1027
1028
1029
1030
1031
1032
1033
1034
1035
1036
1037
1038
1039
1030
1031
1032
1033
1034
1035
1036
1037
1038
1039
1040
1041
1042
1043
1044
1045
1046
1047
1048
1049
1040
1041
1042
1043
1044
1045
1046
1047
1048
1049
1050
1051
1052
1053
1054
1055
1056
1057
1058
1059
1050
1051
1052
1053
1054
1055
1056
1057
1058
1059
1060
1061
1062
1063
1064
1065
1066
1067
1068
1069
1060
1061
1062
1063
1064
1065
1066
1067
1068
1069
1070
1071
1072
1073
1074
1075
1076
1077
1078
1079
1070
1071
1072
1073
1074
1075
1076
1077
1078
1079
1080
1081
1082
1083
1084
1085
1086
1087
1088
1089
1080
1081
1082
1083
1084
1085
1086
1087
1088
1089
1090
1091
1092
1093
1094
1095
1096
1097
1098
1099
1090
1091
1092
1093
1094
1095
1096
1097
1098
1099
1100
1101
1102
1103
1104
1105
1106
1107
1108
1109
1100
1101
1102
1103
1104
1105
1106
1107
1108
1109
1110
1111
1112
1113
1114
1115
1116
1117
1118
1119
1110
1111
1112
1113
1114
1115
1116
1117
1118
1119
1120
1121
1122
1123
1124
1125
1126
1127
1128
1129
1120
1121
1122
1123
1124
1125
1126
1127
1128
1129
1130
1131
1132
1133
1134
1135
1136
1137
1138
1139
1130
1131
1132
1133
1134
1135
1136
1137
1138
1139
1140
1141
1142
1143
1144
1145
1146
1147
1148
1149
1140
1141
1142
1143
1144
1145
1146
1147
1148
1149
1150
1151
1152
1153
1154
1155
1156
1157
1158
1159
1150
1151
1152
1153
1154
1155
1156
1157
1158
1159
1160
1161
1162
1163
1164
1165
1166
1167
1168
1169
1160
1161
1162
1163
1164
1165
1166
1167
1168
1169
1170
1171
1172
1173
1174
1175
1176
1177
1178
1179
1170
1171
1172
1173
1174
1175
1176
1177
1178
1179
1180
1181
1182
1183
1184
1185
1186
1187
1188
1189
1180
1181
1182
1183
1184
1185
1186
1187
1188
1189
1190
1191
1192
1193
1194
1195
1196
1197
1198
1199
1190
1191
1192
1193
1194
1195
1196
1197
1198
1199
1200
1201
1202
1203
1204
1205
1206
1207
1208
1209
1200
1201
1202
1203
1204
1205
1206
1207
1208
1209
1210
1211
1212
1213
1214
1215
1216
1217
1218
1219
1210
1211
1212
1213
1214
1215
1216
1217
1218
1219
1220
1221
1222
1223
1224
1225
1226
1227
1228
1229
1220
1221
1222
1223
1224
1225
1226
1227
1228
1229
1230
1231
1232
1233
1234
1235
1236
1237
1238
1239
1230
1231
1232
1233
1234
1235
1236
1237
1238
1239
1240
1241
1242
1243
1244
1245
1246
1247
1248
1249
1240
1241
1242
1243
1244
1245
1246
1247
1248
1249
1250
1251
1252
1253
1254
1255
1256
1257
1258
1259
1250
1251
1252
1253
1254
1255
1256
1257
1258
1259
1260
1261
1262
1263
1264
1265
1266
1267
1268
1269
1260
1261
1262
1263
1264
1265
1266
1267
1268
1269
1270
1271
1272
1273
1274
1275
1276
1277
1278
1279
1270
1271
1272
1273
1274
1275
1276
1277
1278
1279
1280
1281
1282
1283
1284
1285
1286
1287
1288
1289
1280
1281
1282
1283
1284
1285
1286
1287
1288
1289
1290
1291
1292
1293
1294
1295
1296
1297
1298
1299
1290
1291
1292
1293
1294
1295
1296
1297
1298
1299
1300
1301
1302
1303
1304
1305
1306
1307
1308
1309
1300
1301
1302
1303
1304
1305
1306
1307
1308
1309
1310
1311
1312
1313
1314
1315
1316
1317
1318
1319
1310
1311
1312
1313
1314
1315
1316
1317
1318
1319
1320
1321
1322
1323
1324
1325
1326
1327
1328
1329
1320
1321
1322
1323
1324
1325
1326
1327
1328
1329
1330
1331
1332
1333
1334
1335
1336
1337
1338
1339
1330
1331
1332
1333
1334
1335
1336
1337
1338
1339
1340
1341
1342
1343
1344
1345
1346
1347
1348<br

Output 7:

2) Enter n: 5

Search the elements: 10, 11, 12, 13, 14 for minimum

10

11

12

101 = 8 min

13

121 = 9 min

14

Search or get a value in position 14 giving
enter the key element: 13 (02, 04) for next
element found at 4 (15) giving

Output 8:

(01, 10, "01") 1. 10 - maximum

3) Read n: 5

Enter the elements of the array: 1, 2, 3, 4, 5

1

2

3

4

5

62 maximum no. in array

1 minimum no. in array

C:\ 01 - program search and sort on array
0 21 - program search and sort on array

11/12/23

#include <stdio.h>

void swap(int *a, int *b);

void main()

{

int a = 10;

int b = 45;

printf("value of a and b before swapping = %d %d", a, b);

swap(&a, &b);

printf("swapping of two values after call by reference = %d %d", a, b);

}

void swap(int *a, int *b)

{

int temp;

temp = a;

*a = *b;

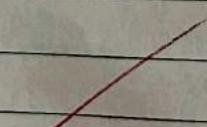
*b = temp;

}

Output:

value of a and b before swapping = 10 45

the values of a and b after swapping = 45 10



classmate
Date _____
Page _____

11/12/23

#include <

#include <

void main()

{

int *p, *q;

int n;

int i;

freopen(

'scanf'

'f = fopen

'fscanf'

'for(i = 0;

'i < n; i++)'

'{ p = q;

'q = fget

'scanf(f,

'*p);'

'*p = q;'

'}');'

'cout << *p;

'}');'

'}');'

'}');'

'}');'

'}');'

'}');'

'}');'

'}');'

'}');'

'}');'

'}');'

'}');'

'}');'

21/7/23

```
#include <stdio.h>
#include <stdlib.h>
void main()
{
    int *p, *q;
    int n;
    int i;
    printf("read n");
    scanf("%d", &n);
    p = (int *) malloc(n * sizeof(int));
    printf("Enter %d elements", n);
    for(i = 0; i < n; i++)
    {
        scanf("%d", p + i);
    }
    q = (int *) calloc(n, sizeof(int));
    printf("Enter seven elements");
    for(i = 0; i < 7; i++)
    {
        scanf("%d", q + i);
    }
    p = realloc(p, 7 * sizeof(int));
    printf("Enter seven elements");
    for(i = 0; i < 7; i++)
    {
        scanf("%d", p + i);
    }
    free(p);
    free(q);
}
```

21/12/23

Output:

scanf n: 3

enter 3 elements: 2

10

3

4

enter 3 elements:

14

5

2

enter seven elements: 2

6

2

8

9

3

4

<20 int> >100 int
<20 double> >100 double
() array 100IP * j + i
D + i

("name") given

("no") given

((int) price * n) given (* 100)

((char) name[0] - '0') given

(a - '0' - '0' - '0') given

(a - '0' - '0' - '0' - '0') given

(a - '0' - '0' - '0' - '0' - '0') given

(a - '0' - '0' - '0' - '0' - '0' - '0') given

(a - '0' - '0' - '0' - '0' - '0' - '0' - '0') given

(a - '0' - '0' - '0' - '0' - '0' - '0' - '0' - '0') given

(a - '0' - '0' - '0' - '0' - '0' - '0' - '0' - '0' - '0') given

(a - '0' - '0' - '0' - '0' - '0' - '0' - '0' - '0' - '0' - '0') given

(a - '0' - '0' - '0' - '0' - '0' - '0' - '0' - '0' - '0' - '0' - '0') given

(a - '0' - '0' - '0' - '0' - '0' - '0' - '0' - '0' - '0' - '0' - '0' - '0') given

(a - '0' - '0' - '0' - '0' - '0' - '0' - '0' - '0' - '0' - '0' - '0' - '0' - '0') given

(a - '0' - '0' - '0' - '0' - '0' - '0' - '0' - '0' - '0' - '0' - '0' - '0' - '0' - '0') given

(a - '0' - '0' - '0' - '0' - '0' - '0' - '0' - '0' - '0' - '0' - '0' - '0' - '0' - '0' - '0') given

(a - '0' - '0' - '0' - '0' - '0' - '0' - '0' - '0' - '0' - '0' - '0' - '0' - '0' - '0' - '0' - '0') given

(a - '0' - '0' - '0' - '0' - '0' - '0' - '0' - '0' - '0' - '0' - '0' - '0' - '0' - '0' - '0' - '0' - '0') given

(a - '0' - '0' - '0' - '0' - '0' - '0' - '0' - '0' - '0' - '0' - '0' - '0' - '0' - '0' - '0' - '0' - '0' - '0') given

(a - '0' - '0' - '0' - '0' - '0' - '0' - '0' - '0' - '0' - '0' - '0' - '0' - '0' - '0' - '0' - '0' - '0' - '0' - '0') given

(a - '0') given

(a - '0') given

(a - '0') given

#include <

#include <

#define m

int stop;

unsigned max

void fresh

{}

uj (top

{}

} given

{

else

{}

top =

& stop

{}

{(m

void pa

{}

int u

{if (t

{given

{}

else

{}

value

jof

fresh

{}

3

01/12/23

CLASSMATE

Date _____

Page _____

```
#include <stdio.h>
#include <stdlib.h>
#define max 5
int top = -1;
int s[max];
void push(int value) {
    if (top == max - 1)
        printf("Stack overflow can't push.");
    else
        {
            top = top + 1;
            s[top] = value;
        }
}
void pop() {
    int value;
    if (top == -1)
        printf("Stack underflow");
    else
        {
            value = s[top];
            top = top - 1;
            printf("Top is popped", value);
        }
}
```

Output :

Enter a n
Enter a n
Enter a n
stack in
stack cl
20 is p
30 is p
10 is p
stack ob
stack ob

S.P.
S.P.
21/10/11

```
void display()
{
```

```
    if (top == -1)
        printf ("Stack is empty underflow");
    else
        printf ("Stack elements are : ");
    for (int i = 0; i < top; i++)
        printf ("%d\t", s[i]);
}
```

```
void display main()
```

```
{
```

```
int value;
```

```
int num;
```

```
printf ("Enter a number");
```

```
scanf ("%d", &num); push(num);
```

```
printf ("Enter a number");
```

```
scanf ("%d", &num);
```

```
push(num);
```

```
printf ("Enter a number");
```

```
scanf ("%d", &num);
```

```
push(num);
```

```
display();
```

```
push
```

```
pop();
```

```
pop();
```

```
pop();
```

```
display();
```

Output :

Enter a no: 10

Enter a no: 20

Enter a no: 30

Stack is overflow can't push 10, 20, 30

Stack elements are: 10, 20, 30

20 is popped

30 is popped

10 is popped

Stack elements are: 10, 10 is popped

Stack is underflow: can't pop

S.P.T
21/12/23

20/12/23

Evaluation of Infix to Postfix

classmate
Date _____
Page _____ 10

```
① #include <iostream.h>
#include <string.h>
#include <limits.h>
#define MAX 100
char stack [MAX];
char infix [MAX];
char postfix [MAX];
char *postfix (max);
int top = -1;
void push (char);
char pop ();
int isEmpty ();
void infixToPost ();
void print ();
int freeCalc (char);
int main ()
{
    printf ("Enter the Infix expression ");
    gets (infix);
    infixToPost ();
    print ();
    return 0;
}
```

```
void infixToPost ()
{
    int i, j = 0;
    char symbol, next;
    for (i = 0; i < strlen (infix); i++)
    {
        symbol = infix[i];
        switch (symbol)
        {
            case 'c':
                break;
            default:
```

```
                if (symbol == '+' || symbol == '-')
                    postfix[j] = symbol;
                    j++;
                else if (symbol == '*' || symbol == '/')
                    while (!isEmpty () && precedence (postfix[j]) >= precedence (symbol))
                        postfix[j] = pop ();
                    postfix[j] = symbol;
                    j++;
                else if (symbol == '(')
                    push (symbol);
                else if (symbol == ')')
                    while (top != -1 && infix[top] != '(')
                        postfix[j] = pop ();
                    top--;
                    j++;
                else
                    postfix[j] = symbol;
                    j++;
            }
        }
    }
}
```

```
case ')':
    while (precedence (postfix[j]) >= precedence ('('))
        postfix[j] = pop ();
    j++;
    break;
case '-':
    case '*':
    case '/':
    case '^':
        postfix[j] = symbol;
        j++;
        break;
    default:
        if (precedence (postfix[j]) >= precedence ('^'))
            postfix[j] = symbol;
            j++;
        else
            break;
```

```
while (i < n)
    if (infix[i] == '^')
        postfix[j] = infix[i];
        j++;
    else if (precedence (infix[i]) >= precedence (postfix[j]))
        postfix[j] = infix[i];
        j++;
    else
        break;
```

```
int freeCalc ()
{
    switch (symbol)
    {
        case '+':
        case '-':
        case '*':
        case '/':
            if (top == -1)
                cout << "Error";
            else
                result = result + pop ();
            break;
        case '^':
            if (top == -1)
                cout << "Error";
            else
                result = pow (pop (), pop ());
            break;
        default:
            cout << "Error";
```

```
cout << result;
```

```
cout << endl;
```

```
cout << endl;
```

```
defau
```

case 'c':

while (cnext == prop(c) == 'c')

postfix[i++ == next];

break;

case 'r':

case '-':

case '*':

case '/':

case '^':

while (!isEmpty() && precedence(stack.top())

>= precedence(symbol))

postfix[i++ == prop());

push(symbol);

break;

default:

postfix[i++ == symbol];

}

while (!isEmpty())

postfix[i++ == prop());

postfix[i] == '0';

int precedence(char symbol)

switch (symbol)

case '^':

return 3;

case ')':

case '!':

return 2;

case '+':

case '-':

return 1;

default:

return 0;

void print()

{ init i=0;

for(i=0; i < stack_size; i++) cout << stack[i] << " ";

} fixin (" . . c ", postfix[i++]);

} fixin (" In ");

} void push (char c)

{ if (top == MAX - 1)

cout << "Stack overflow";

return;

} top++;

stack [top] = c;

} char pop()

{ char c;

if (top == -1)

{ cout << "Stack underflow";

exit (1);

} c = stack [top];

top = top - 1;

return c;

exit isEmpty()

{

if (top == -1) return 1;

else

return 0;

}

Output:

Enter the infix
the equivalent

exit isEmpty()

{

if (top == -1)

return 1;

else

return 0;

{

Output:

Enter the infix expression: (a * b) + (c * d)

The equivalent postfix expression is: ab * cd * +

11/1/20

Queue

```
② #include < stdio.h >  
#include < stdlib.h >  
#define MAX 6  
int A[MAX];  
int front = -1;  
int rear = -1;  
void insert(int x) {  
    if (rear == MAX - 1)  
    {
```

```
        printf("Queue overflow");  
        exit(1);  
    }  
    if (front == -1) {  
        front = rear = 0;  
    }  
    else {  
        rear++;  
    }  
    A[rear] = x;  
}
```

```
void delete() {  
    if (front == -1)
```

```
        printf("Queue underflow");  
        exit(1);  
}
```

```
    printf("Deleted elements are %d\n", A[front]);  
    front++;  
    if (front > rear)
```

```
        front = rear = -1;  
    }
```

Page 4

```
void display()  
{  
    if (front == -1)  
    {  
        printf("Queue Underflow");  
        exit(1);  
    }  
    int i;  
    for (i = front; i <= rear; i++)  
    {  
        printf("%d ", A[i]);  
    }  
    printf("\n");  
}  
void main()  
{  
    int choice;  
    do {  
        printf("1. Insert\n");  
        printf("2. Delete\n");  
        printf("3. Display\n");  
        printf("4. Exit\n");  
        printf("Enter your choice: ");  
        scanf("%d", &choice);  
        switch (choice) {  
            case 1:  
                insert();  
                break;  
            case 2:  
                delete();  
                break;  
            case 3:  
                display();  
                break;  
            case 4:  
                exit(0);  
            default:  
                printf("Invalid choice\n");  
        }  
    } while (choice != 4);  
}
```

void display()

{

if (front == -1)

{

cout << ("Queue underflow");

exit(1);

}

int i;

cout << ("front to rear: ");

for (i = front; i < rear; i++)

{

cout << (*rd, QSI);

}

freopen ("In", "r", stdin);

void main()

{

int choice;

int x;

do {

cout << ("Queue Operations");

cout << ("1. Insertion");

cout << ("2. deletion In");

cout << ("3. display In");

cout << ("4. exit In");

cout << ("Enter the choice");

scanf ("%d", &choice);

switch(choice)

{

case 1:

freopen ("Enter the element to be inserted");

scanf ("%d", &x);

insert(x);

break;

case 2: delete();

break;

case 3: display();
break;

case 4: exit(1);

default: printf("invalid choice try again");

{ while (choice != 4);

}

Output:

Queue operation:

1. insertion

2. deletion

3. display

4. exit

enter the choice 1

enter the element to be inserted 34

Queue operation

1. insertion

2. deletion

3. display

4. exit

enter the choice 2

enter the element to be deleted 34

Queue operation

1. insertion

2. deletion

3. display

4. exit

enter the choice 4

Date _____
Page _____

Postfix eval

② #include <cs.h>

#include <iostream.h>

#include <stack.h>

#define max 10

int stack;

int top = -1;

void push(int item);

int pop();

int front();

int return();

3

void main()

char ch;

int yes;

for (ch = ' ',

if (ch == 'c')

else if (ch == 'd')

else if (ch == 'p')

else if (ch == 'f')

else if (ch == 'q')

else if (ch == 's')

else if (ch == 'e')

else if (ch == 'r')

else if (ch == 'm')

else if (ch == 'l')

else if (ch == 'n')

else if (ch == 'o')

else if (ch == 'v')

else if (ch == 'w')

else if (ch == 'x')

else if (ch == 'y')

else if (ch == 'z')

else if (ch == 't')

else if (ch == 's')

else if (ch == 'd')

else if (ch == 'c')

else if (ch == 'e')

else if (ch == 'f')

else if (ch == 'g')

else if (ch == 'h')

else if (ch == 'i')

else if (ch == 'j')

else if (ch == 'k')

else if (ch == 'l')

else if (ch == 'm')

CLASSMATE
Date _____
Page _____
11/01/29
week - 4

CLASSMATE
Date _____
Page _____
19

```
#include <stdio.h>
#include <stdlib.h>
#define size 50
int queue[50];
int queueSize;
int rear = -1;
int front = -1;
int isFull()
```

```
{ if (front == (rear + 1) % size)
```

```
    return 0;
```

```
}
```

```
else
```

```
{
```

```
    return -1;
```

```
}
```

```
}
```

```
int isEmpty()
```

```
{
```

```
if (front == -1 && rear == -1)
```

```
    return 0;
```

```
}
```

```
else
```

```
{
```

```
    return -1;
```

```
}
```

```
}
```

```
void Enqueue(int)
```

```
{ int item;
```

```
if (!isFull() == 0)
```

```
{
```

```
    printf("Queue overflow\n");
```

```
}; return;
```

clr
{
if (IsEmpty() == 0)
front = 0;
rear = 0;

clr
{
xcar = (rear + 1) % size;
Q(xcar) = x;

int Dequeue()
{

clr
{
if (IsEmpty() == 0)
faring ("Queue underflow");

clr
{
if (front == rear)

x = queue(front);
front = -1;
rear = -1;

else
2

x = queue(front);
front = (front + 1) % size;

return x;

void display()
2

clr v;
if (IsEmpty())
2

faring ("Queue underflow");
else
2

front = 0;

for (i = 0;
i < size;
i++)

for (j = front;
j < rear;
j++)

front++;

void main()
{

int choice;
while (choice != 5)

choice =

void display()

{ unit ';

if (isempty() == 0)

{ printf ("Queue is empty\n");

else

{

forint ("Queue elements : \n").

for (i = front; i != rear; i = (i + 1) % size)

{ printf ("%d\n", queue[i]);

printf ("%.1d\n", queue[r]);

void main()

{ int i, j, k, l, m, n, o, p, q, r, s, t, u, v, w, x, y, z;

unit choice, x, y, z;

while (1); b & c & d & e & f & g & h & i & j & k & l & m & n & o & p & q & r & s & t & u & v & w & x & y & z;

{

printf ("1.Enqueue , 2.Dequeue 3.display 4.exit

choice ("Enter your choice"),

scanf ("%d", &choice),

switch (choice)

{

case 1: printf ("Enter the number to be inserted\n");

scanf ("%d", &x);

enqueue(x);

break;

case 2: b = dequeue();

printf ("%.1d has removed from the
queue\n", b);

break;

case 3: display();
break;

case 4: exit(1);

default: printf("Invalid choice");

{

}

)

Output

1.Enqueue 2.dequeue 3.display 4.exit

Enter your choice : 1

Enter the number to be inserted

23

1.Enqueue 2.Dequeue 3.display 4.exit

Enter the number to be inserted

45

1.Enqueue 2.Dequeue 3.display 4.exit

Enter the number to be inserted

28

1.Enqueue 2.dequeue 3.display 4.exit

Enter the choice : 2

Enter the no. to be deleted

45

Enter your choice : 3: Queue elements

23 28

Enter your choice 4

11/01/24 11:45 AM
Date
Page 14

Week - 4

11/01/24 11:45 AM

#include < std

#include < st

struct Node

{

int data

struct N

};

struct No

};

struct No

{size of

if (new

);

print

exit(

);

newNode

newNo

return

);

struct N

ceil val

};

struct

newNo

getdata

);

struct

newNo

getdata

);

struct

newNo

getdata

);

Week - 4
11/01/24 (Create linked list)

#include <stdio.h>

#include <stdlib.h>

struct Node

{

int data;

struct Node *next;

}

struct Node *createNode(int value)

{

struct Node *new = (struct Node *) malloc

(sizeof(struct Node));

if (newNode == NULL)

{

fprintf("Memory allocation failed in ");

exit(1);

}

newNode->data = value;

newNode->next = NULL;

return newNode;

}

struct Node *insertAtBeginning (struct Node *head,
int value)

{

struct Node *newNode = createNode(value);

newNode->next = head;

return newNode;

}

struct Node *insertAtAnyPos (struct Node *head,
int value, int pos)

{

struct Node *newNode = createNode(value);

struct Node *temp = head;

if (pos == 1) {

newNode->next = head;

return newNode;

3
int i;
for (i=1; i < pos-1; i++)

{
temp = temp->next;

}
newNode->next = temp->next;

temp->next = newNode;

return head;

}{
struct Node * insertAtEnd (struct Node * head,
int value)

{
struct Node * newNode = createNode (value);

if (head == NULL) {
return head;

}
return newNode;

}{
struct Node * temp, * head;

while (temp != NULL)

{
temp = temp->next;

}
temp->next = newNode;

return head;

}{
void displayList (struct Node * head)

{
struct Node * temp = head;

while (temp != NULL)

{
printf (" %d ", temp->data);

temp = temp->next;

}
printf ("\n");

}{

unit main()

"struct node * head = NULL;

int choice, value, pos;

while(1)

{

'print ("1. Insert at end").

'print ("2. Insert at beginning (0)").

'print ("3. Insert at any position (n)").

'print ("4. Display list (n)").

'print ("5. Exit (n)").

'print ("Enter your choice:").

scanf ("%d", &choice);

switch(choice)

{

case 1: print ("Enter a new value to be inserted").

scanf ("%d", &value);

head = insertAtEnd(head, value);

break.

case 2: print ("Enter the value to be inserted").

scanf ("%d", &value);

head = insertAtBeginning(head, value);

break.

case 3: print ("Enter the value to be inserted").

scanf ("%d", &value);

print ("Enter the position").

scanf ("%d", &pos);

head = insertAtAnyPos(head, value, pos);

break.

case 4: displayList(head);

break.

case 5: exit(0);

default: print ("Invalid choice")

}

return 0;

18/01/24 Week - 5

Output

1. Insert at end
2. Insert at beginning
3. Insert at any position
4. Exit

Enter your choice: 2

Enter the value to be inserted: 12

Enter your choice: 2

Enter the value to be inserted: 13

Enter your choice: 4

13 → 12 → NULL

Enter your choice: 1

Enter the value to be inserted: 23

Enter your choice: 2

13 → 12 → 23 → NULL

Enter your choice: 3

Enter the value to be inserted: 12

Enter the position to insert: 3

Enter your choice: 4

13 → 12 → 12 → 23 → NULL

#include <stdio.h>

#include <stdlib.h>

struct Node

{

int data;

struct Node*

};

struct node

void insert()

void begin()

void end()

void search()

void display()

void main()

{

main()

while (C)

{

print

switch

{

case

case

case

18/10/22 week - 5

```
#include <stdio.h>
#include <stdlib.h>
struct Node
{
    int data;
    struct Node * next;
};

struct Node * head=NULL;

void insert();
void begin();
void end();
void array();
void display();
int main()
```

{

```
int choice;
while (c != 1)
```

{

```
    printf("1. Insert elements\n");
    printf("2. Delete at the beginning\n");
    printf("3. Delete at the end\n");
    printf("4. Delete at any position\n");
    printf("5. Display\n");
    printf("6. Exit\n");
    printf("Enter your choice: ");
    scanf("%d", &choice);
    switch (choice)
```

{

```
    case 1: insert();
              break;
```

```
    case 2: begin();
              break;
```

```
    case 3: end();
              break;
```

case 4 : at any pos();
break;

case 5 : display();
break;

case 6 : exit(0);

default :

 printf("Entered choice In");

{

{

 return 0;

}

void insert()

{

 struct Node *ftr, *temp;

 int data;

 ftr = (struct Node *) malloc(sizeof(struct Node));

 if (ftr == NULL)

{

 printf("Memory allocation failed\n");

 exit(1);

}

 printf("Enter data to insert : ");

 scanf("%d", &data);

 ftr->data = data;

 ftr->next = NULL;

 if (head == NULL)

{

 head = ftr;

 printf("Node inserted successfully\n");

}

else

{

 temp = head;

 while (temp->next != NULL)

2

3 temp

temp → n

freemp C

3

void beg

3

struct

if (head

s

3 free

else

ftr =

head;

free (

3 free

3 free

3 void c

3

struct

if (head

3 free

3 else

ftr whi

3 fr

3 fr

```
2
    } temp = temp->next;
    } temp->next = fib;
    printf ("Node inserted successfully");
}
```

```
3 void begin()
```

```
{ struct Node *ptr;
if (head == NULL)
```

```
} printf ("list is empty");
```

```
else
```

```
    fib = head;
```

```
    head = fib->next;
```

```
    free (ptr);
```

```
} printf ("node deleted at the beginning\n");
```

```
}
```

```
void end()
```

```
{
```

```
struct Node *ptr, *prev;
```

```
if (head == NULL)
```

```
{
```

```
} printf ("list is empty\n");
```

```
else
```

```
    fib = head;
```

```
    while (fib->next != NULL)
```

```
{
```

```
    prev = fib;
```

```
    fib = fib->next;
```

```
}
```

if ($fptr == \text{head}$)

{
 $\text{head} = \text{NULL}$;
}

else

{
 $\text{prev} \rightarrow \text{next} = \text{NULL}$;
}

$\text{free}(\text{fptr})$;

printf ("Node deleted from the end\n");

void $\text{anypos}()$

{
 struct Node *fptr, *prev;

 int loc, i = 1;

 printf ("Enter the position: ");

 scanf ("%d", &loc);

 if ($\text{head} == \text{NULL}$)

{
 }

 printf ("list is empty\n");

 return;

}

 fptr = head;

 if ($loc == 1$)

{
 }

 head = fptr -> next;

 free (fptr);

 printf ("Node deleted from position 1\n");

 return;

}

while ($fptr !=$

{
 $\text{prev} = \text{fptr}$;

$\text{fptr} = \text{fptr}$

while (ptr != NULL & & p < loc)

{

prev = ptr;

ptr = ptr->next;

i++;

{

if (ptr == NULL)

{

printf ("Position %d is out of bounds\n", loc);

{

else

{

prev->next = ptr->next;

free (ptr);

printf ("Node deleted from position %d\n", loc);

{

void display()

{

struct Node *ptr;

if (head == NULL)

{

printf ("List is empty");

{

else

{

ptr = head;

while (ptr != NULL)

{

printf ("%d ", ptr->data);

ptr = ptr->next;

{

printf ("\n");

{

Output

Menu -

1. Insert elements
2. Delete at the beginning
3. Delete at the end
4. Delete at any position
5. Display
6. Exit

Enter your choice : 1

Enter data to insert : 12

Enter your choice : 1

Enter data to insert : 13

Enter your choice : 1

Enter data to insert : 14

Enter your choice : 1

Enter data to insert : 15

Enter your choice : 5

12 13 14 15

Enter your choice : 1

Node deleted from beginning

Enter your choice : 5

13 14 15

Enter your choice : 3

Node deleted at the end

Enter your choice : 5

13 14

Enter your
Enter the position
Position 2

Enter your
3

deetcode

```
typedef struct {  
    int size;  
    unit_top;  
    int *s;  
    unit *minstack;  
} minstack;
```

```
minstack * minstackCreate() {
    minstack *st = (minstack *) malloc(sizeof(minstack));
    if(st == NULL)
        fprintf("Memory allocation failed");
    exit(0);
}
```

$$\Delta t \rightarrow \Delta x = 5 :$$

$$at \rightarrow stop = -1$$

$\text{at} \rightarrow s = (\text{int} *) \text{malloc}(\text{at} \rightarrow \text{size} * \text{size of } (\text{int}))$.
 $\text{at} \rightarrow \text{minstack} = (\text{int} *) \text{malloc}(\text{at} \rightarrow \text{size} * \text{size of } (\text{int}))$;
 if ($\text{at} \rightarrow s == \text{NULL}$)
 {

printf ("memory allocation failed").
free (st → s);

...and the world was created.

```
free(st->memstack);  
exit(0);
```

3

return st.

```
void minstackpush(minstack* obj, int val) {  
    if (obj->top == obj->size - 1)  
        S
```

going ("stack is overflow");

else {

~~OK → stop it~~

$\text{obj} \rightarrow \mathcal{S}(\text{obj} \rightarrow \text{top}) = \text{value}$;

$\text{obj}(\text{obj} \rightarrow \text{Top} = 0 \quad || \quad \text{val} < \text{obj} \rightarrow \text{minstack}$
 $(\text{obj} \rightarrow \text{Top} = 1) \}$

$(\partial y \rightarrow \text{Topo})$

$O(j) \rightarrow$ minstack $\{O(j) \rightarrow \text{top}\} = \text{val};$

3 click

$\text{obj} \rightarrow \text{minstack}$ (obj \rightarrow dup) \rightarrow obj \rightarrow minstack

$\{0\} \rightarrow \{0\} - 1\}$;

9

void minStackPush(minStack *obj)

5

unit value

if ($\text{obj} \rightarrow \text{tch} = -1$)

3

foreint ("underflow").

cls

3

`value:obj → (S{obj} → top)`;

obj' → stop -')

obj → prop-
function ("obj is justified in " value);

int minStackPush(minStack *obj){

{
 unit value = -1;

 if (obj->top == -1) {

 fixing ("underflow in");

 exit(0);

}
else {

 value = obj->obj->top;];

 return value;

 } top

 unit value = -1;

 if (obj->top == -1) {

 fixing ("underflow in");

 exit(0);

}
else {

 value = obj->obj->top;];

 return value;

int minStackGetMin (minStack *obj) {

 if (obj->top == -1) {

 fixing ("underflow in");

 exit(0);

else

{

 return obj->

 }

void minDel (

{

 free (

 free (

 free (

y

else

{

}

}

}

}

}

}

}

}

}

}

}

}

}

}

}

}

}

}

}

}

}

}

}

}

}

}

}

}

}

}

}

}

}

}

}

}

}

}

}

}

}

}

}

}

}

}

}

}

}

}

}

}

}

}

}

}

}

}

}

}

}

}

}

}

}

}

}

}

}

}

}

}

}

}

}

}

}

}

}

}

}

}

}

}

}

}

}

}

}

}

}

}

}

}

}

}

}

}

}

}

}

}

}

}

}

}

}

}

}

}

}

}

}

}

}

}

}

}

}

}

}

}

}

}

}

}

}

}

}

}

}

}

}

}

}

}

}

}

}

}

}

}

}

}

}

}

}

}

}

}

}

}

}

}

}

}

}

}

}

}

}

}

}

}

}

}

}

}

}

}

}

}

}

}

}

}

}

}

}

}

}

}

}

}

}

}

}

}

}

}

}

}

}

}

}

}

}

}

}

}

}

}

}

}

}

}

}

}

}

}

}

}

}

}

}

}

}

}

}

}

}

}

}

}

}

}

}

}

}

}

}

}

}

}

}

}

}

}

}

}

}

}

}

}

}

}

}

}

}

}

}

}

}

}

}

}

}

}

}

}

}

}

}

}

}

}

}

}

}

}

}

}

}

}

}

}

}

}

}

}

}

}

}

}

}

}

}

}

}

}

}

}

}

}

}

}

}

}

}

}

}

}

}

}

}

}

}

}

}

}

}

}

}

}

}

}

}

}

}

}

}

}

}

}

}

}

}

}

}

}

}

}

}

}

}

}

}

}

}

}

}

}

}

}

}

}

}

}

}

}

}

}

}

}

}

}

}

}

}

}

}

}

}

}

}

}</div

25/01/24

Week - 6 : linked list : insert, Reverse, Concatenation

```
#include <stdio.h>
#include <stdlib.h>
struct Node {
    int data;
    struct Node * next;
};
```

```
struct Node * insertNode(struct Node * head,
                        int newData) {
```

```
    struct Node * new_node = (struct Node *) malloc
        (sizeof(struct Node));
    new_node->data = newData;
```

```
    new_node->next = head;
    return new_node;
```

```
struct Node * bubbleSort(struct Node * head)
```

```
{ if (head == NULL || head->next == NULL)
    return head;
```

```
    struct Node * temp;
    struct Node * end = NULL;
    do {
```

```
        swapped = 0;
```

```
        temp = head;
```

```
        while (temp->next != end)
```

```
        {
```

```
            if (temp->data > temp->next->data)
```

```
                int tempData = temp->data;
```

```
                temp->data = temp->next->data;
```

```
                temp->next->data = tempData;
```

```
                swapped = 1;
```

```
}
```

temp = temp → next;

{

end = temp;

{ while (temp ≠ NULL);

return head;

{

Struct Node * severesel (Struct Node * head)

{

Struct Node * prev = NULL;

Struct Node * current = head;

Struct Node * next = NULL;

while (current != NULL)

{

next = current → next;

current → next = prev;

prev = current;

current = next;

{

return prev;

{

Struct Node * concatlist (Struct Node * head1,

Struct Node * head2)

{

if (head1 == NULL)

{

return head2;

{

Struct Node * temp = head1;

while (temp != NULL)

{

printf ("%d", temp → data);

temp = temp → next;

printf ("\n");

{

exit main()

{

Struct Node

exit choice,

while (c)

{

freeif (c

scanf (

switch (

case 1 :

free

scanf

free

scanf

free

scanf

case 2 :

free

scanf

case 3 :

case 4 :

```
int main()
```

```
{
```

```
    struct Node * head = NULL;
```

```
    int choice, data;
```

```
    while(1)
```

```
{
```

```
    printf("1. Insert node in a. do it linkedlist\n"
           "2. Reverse linked list\n"
           "3. Concatenate\n"
           "4. Print linked list\n"
           "5. Exit\n"
           "6. Exit\n");
```

```
    printf("Enter your choice ");
```

```
    scanf("%d", &choice);
```

```
    switch(choice){
```

```
        case 1:
```

```
            printf("Enter the data to be inserted: ");
```

```
            scanf("%d", &data);
```

```
            head = insertNode(head, data);
```

```
            break;
```

```
        case 2: head = bubbleSort(head);
```

```
            printf("linked list sorted\n");
```

```
            break;
```

```
        case 3: head = reverseList(head);
```

```
            printf("linked list reversed\n");
```

```
            break;
```

```
        case 4: printf("Enter the data for the second\n"
                         "linked list ");
```

```
        struct Node * head2 = NULL;
```

```
        while(1){
```

```
            scanf("%d", &data);
```

```
            if(data == -1)
```

```
                break;
```

```
            head2 = insertNode(head2, data);
```

```
}
```

```
        head = concatList(head, head2);
```

freeif ("linked lists concatenated\n"),
break;
case 5: freeList (head);

case 6: exit (0);
default: printf ("Invailed choice\n");

}

return 0;

}

Output:

1. InsertNode
2. Sortdlinked list
3. Reversed linked list
4. Concatenate linked list
5. Print linked list
6. Exit

Enter your choice : 1

Enter the data to insert : 11

Enter your choice : 1

Enter the data to insert : 10

Enter your choice : 1

Enter the data to insert : 15

Enter your choice : 1

Enter the data to insert : 18

Enter your choice : 2

10 14 15

list is sorted

Enter your choice : 5

10 14 16 18

Enter your choice
dlinked list

Enter your choice
18 15 14 11

Enter your choice
Enter the data
12 9 8 7

Enter your choice
18 15 14

Enter your choice
12 11 8 7 6 5 4 3 2 1

Enter your choice

Enter your choice: 3
linked list is reversed

Enter your choice: 5
18 15 14 10

Enter your choice: 4
Enter the data for second linked list
12 9 8 7 1

Enter your choice: 5
18 15 14 10 12 9 8 7

Enter your choice: 6
*Entered
12
15
14
10
18*

stack implementation

```
#include <stdio.h>
```

```
#include <stdlib.h>
```

```
struct Node {
```

```
    int data;
```

```
    struct Node *next;
```

```
};
```

```
struct Node *createNode(int data)
```

```
{
```

```
    struct Node *newNode, (*struct Node * )
```

```
    malloc (sizeof (struct Node) );
```

```
    newNode->data = data;
```

```
    newNode->next = NULL;
```

```
    return newNode;
```

```
}
```

```
void push (struct Node **top, int data)
```

```
{
```

```
    struct Node *newNode = createNode(data);
```

```
    newNode->next = *top;
```

```
*top = newNode;
```

```
printf (" %d pushed to the stack \n ", data);
```

```
}
```

```
int pop (struct Node **top) {
```

```
    if (*top == NULL)
```

```
{
```

```
    printf (" stack is empty \n " );
```

```
    return -1;
```

```
}
```

```
int topValue = (*top)->data;
```

```
struct Node *temp = *top;
```

```
*top = (*top)->next;
```

```
free (temp);
```

```
return topValue;
```

```
}
```

```
void display()
```

```
{
```

```
    if (*top ==
```

```
{
```

```
    printf (" stack is empty \n " );
```

```
}
```

```
    printf (" stack is full \n " );
```

```
    while (top !=
```

```
    free (temp);
```

```
    top = temp;
```

```
}
```

```
    printf (" stack is empty \n " );
```

```
}
```

```
int main ()
```

```
{
```

```
    struct
```

```
    int choice;
```

```
    while (choice != 5)
```

```
{
```

```
    printf
```

```
    printf
```

```
    scanf
```

```
void displayStack (struct Node *top)
```

{

```
if (top == NULL)
```

{

```
printf ("Stack is empty\n");
```

```
return;
```

{

```
for (top; top != NULL; top = top->next)
```

```
printf ("%d", top->data);
```

```
top = top->next;
```

{

```
printf ("\n");
```

```
int main ()
```

{

```
struct Node *stackTop = NULL;
```

```
int choice, data;
```

```
while (choice)
```

{

```
printf ("1. Push in 2. Pop in 3. DisplayStack\n"
       "4. Exit\n");
```

```
printf ("Enter your choice ");
```

```
scanf ("%d", &choice);
```

```
switch (choice)
```

{

```
case 1: printf ("Enter data to push: ");
scanf ("%d", &data);
```

```
push (stackTop, data);
```

```
break;
```

```
case 2: printf ("Popped element %d\n",
                pop (&stackTop));
```

```
break;
```

```
case 3: displayStack (stackTop);
```

```
break;
```

Date _____
Page _____

case 4 : exit(0);
default : printf ("Invalid choice ");
3

3
3
return 0;

Output:

1. Push
2. Pop
3. Display stack
4. Exit

Enter your choice: 3

Enter data to push: 12

12 pushed to the stack

1. Push
2. Pop
3. Display stack
4. Exit

Enter your choice: 1

Enter data to push: 13

13 pushed to the stack

1. Push
2. Pop
3. Display stack
4. Exit

Enter your choice: 1

Enter data to push: 14

14 pushed to the stack

1. Push
2. Pop
3. Display
4. Exit

Enter co

13 push

1

1. Push

2. Pop

3. Disp

4. Exi

Enter 4

Push

1

1. Push

2. Pop

3. Disp

4. Exi

Enter 4

Push

1

1. Push

2. Pop

3. Disp

4. Exi

- 1. Push
- 2. Pop
- 3. Display Stack
- 4. Exit

Enter your choice: 2 Enter data to push: 15
15 pushed to the stack

- 1. Push
- 2. Pop
- 3. Display Stack
- 4. Exit

Enter your choice: 2
Popped element: 15

- 1. Push
- 2. Pop
- 3. Display Stack
- 4. Exit

Queue implementations

#include <stdio.h>

#include <stdlib.h>

struct Node {

{

int data;

struct Node * next;

}

struct node * head = NULL;

void enqueue()

{

struct node * temp;

temp = (struct node *) malloc(sizeof(struct node));

scanf("Enter value to be inserted: %d",

&temp->data);

temp->next = NULL;

if(head == NULL)

{

head = temp;

return;

}

else

{

struct node * ptr;

ptr = head;

while(ptr->next != NULL)

{

ptr = ptr->next;

ptr->next = temp;

}

void deque()

{

struct node

if(head ==

{

front

return

}

else

{

ptr =

head

front

free

ptr

3

void

{

if (head ==

{

front

return

3

else

{

ptr =

head

front

ptr

3

front

ptr

3

void dequeue()

```
{  
    struct node *ftr;  
    if (frcrd == NULL)  
        }
```

```
    fprintf ("Queue is empty\n");  
    return;  
}
```

```
else  
{
```

```
    ftr = head;
```

```
    head = ftr->next;
```

```
    free (ftr);  
}
```

```
}
```

void display()

```
{  
    if (frcrd == NULL)  
        }
```

```
    fprintf ("Queue is empty\n");  
    return;  
}
```

```
else  
{
```

```
    struct node *ftr;
```

```
    ftr = head;
```

```
    while (ftr != NULL)
```

```
    {  
        fprintf ("%d\t", ftr->data);  
        ftr = ftr->next;  
    }
```

```
}
```

void main ()

{

int choice;
while(1)
{

printf("1.enqueue(n)");
printf("2.dequeue(n)");
printf("3.display()");
printf("4.exit()");

scanf("Enter your choice");

scanf("%d", &choice);

switch(choice)

{ case 1: enqueue();
break;

case 2: dequeue();
break;

case 3: display();
break;

case 4: exit();
break;

}
}

Output

1.enqueue

2.dequeue

3.display

4.exit

Enter yo

enter va

(1)

enter yo

enter va

(2)

enter yo

enter va

(3)

enter yo

enter va

(4)

enter yo

enter va

(5)

enter yo

Value

(6)

enter

Value

(7)

enter

value

(8)

enter

1

Output

1. enqueue
2. dequeue
3. display
4. exit

enter your choice : 1

enter value to be inserted : 12

enter your choice : 2

enter value to be inserted : 54

enter your choice : 3

enter value to be inserted : 19

enter your choice : 4

enter value to be inserted : 20

enter your choice : 1

enter value to be inserted : 30

enter your choice : 3

enter value to be inserted : 12 54 19 20 30

enter your choice : 2

Value dequeued 12

enter your choice : 2

Value dequeued : 54

enter your choice : 3

19 20 30

01/02/24

```
#include <stdio.h>
#include <stdlib.h>
struct Node {
    int data;
    struct Node *prev;
    struct Node *next;
};

struct Node *createNode(int value) {
    struct Node *newNode = (struct Node *) malloc(sizeof(struct Node));
    if (newNode == NULL) {
        fprintf(stderr, "Memory allocation failed\n");
        exit(1);
    }
    newNode->data = value;
    newNode->prev = NULL;
    newNode->next = NULL;
    return newNode;
}
```

```
void insertToFirst (struct Node **head, int value)
```

```
{ struct Node *newNode = createNode(value);
    if (*head == NULL) {
        *head = newNode;
        newNode->next = *head;
        (*head)->prev = newNode;
        *head = newNode;
    } else {
        newNode->next = *head;
        (*head)->prev = newNode;
        *head = newNode;
    }
}
```

```
void delete()
{
    if (*head == NULL)
        return;
    struct Node *cur;
    while (cur != NULL) {
        if (cur->prev == NULL) {
            if (cur->next == NULL) {
                free(cur);
                cur = NULL;
            } else {
                cur = cur->next;
                cur->prev = NULL;
                free(cur);
                cur = NULL;
            }
        } else {
            if (cur->next == NULL) {
                cur->prev->next = NULL;
                free(cur);
                cur = NULL;
            } else {
                cur = cur->next;
                cur->prev = cur->prev->next;
                free(cur);
                cur = NULL;
            }
        }
    }
}
```

```
void deletenode (struct Node ** head, int value)
{
    if (*head == NULL)
    {
        printf ("List is empty\n");
        return;
    }
    struct Node * current = *head;
    while (current != NULL)
    {
        if (current->data == value)
        {
            if (current->prev == NULL)
                current->prev->next = current->next;
            else
                *head = current->next;
            free (current);
            printf ("Node with value %d deleted\n", value);
            return;
        }
        current = current->next;
    }
}
```

```
printf ("Node with value %d deleted not found\n");
```

```
void displalist (struct Node * head)
```

```
{
```

```
if (*head == NULL)
{
    printf ("List is empty\n");
    return;
}
```

```
friend ("Double linked list"),  
{ while (head != NULL)  
    {
```

```
        friend ("• d ↔", head->data),  
        head = head->next;
```

```
    } friend ("NULL\n").
```

```
exit main();
```

```
{
```

```
struct node *head = NULL,
```

```
int choice, value;
```

```
do {
```

```
    friend ("1. Insert To Left | n 2. Delete by  
            3. Display | n 4. Exit | n"),
```

```
friend ("Enter your choice"),
```

```
scanf ("%d", &choice);
```

```
switch (choice) {
```

```
{
```

```
case 1: friend ("Enter the value to insert.");
```

```
scanf ("%d", &value);
```

```
insertToLeft (&head, value);
```

```
break;
```

```
case 2: friend ("Enter the value to delete");
```

```
scanf ("%d", &value);
```

```
deleteNode (&head, value);
```

```
break;
```

```
case 3: displayList (head);
```

```
break;
```

```
case 4: friend ("Exiting the program\n");
```

```
break;
```

```
default: friend ("Invalid choice. Please enter  
          a valid option\n");
```

```
}
```

```
} while (choice != 4). return 0;
```

leetcode: linked

except:

Output:

1 Insert to LC

2 Delete by

3 Display

4 Exit

Enter your c

Enter the v

Enter your

Enter the n

Enter your

Enter val

Enter your

Enter Val

Enter yo

pously

Enter u

Enter t

Node u

Enter u

Row u

linked list

output:

- Output:
1 Insert to left
2 Delete by value
3 Reverse
4 Exit

Enter your choice: 1

Enter the value to insert: 12

Enter your choice: 1

Enter the value to insert: 13

Enter your choice: 1

Enter value to insert: 12

Enter your choice: 1

Enter value to insert: 5

Enter your choice: 3

possibly linked list: $12 \leftrightarrow 13 \leftrightarrow 14 \leftrightarrow \text{NULL}$

Enter your choice: 2

Enter the value: 13

Node with value 13 deleted

Enter your choice: 3

possibly linked list: $12 \leftrightarrow 14 \rightarrow \text{NULL}$

Leetcode : Reversing linked list as tree

```
struct ListNode * reverseBetween (struct ListNode * head, int left, int right) {  
    if (head == NULL || left == right) {  
        return head;  
    }  
}
```

```
struct ListNode * dummy = (struct ListNode *) malloc (sizeof (struct ListNode));  
dummy->next = head;
```

```
struct ListNode * prev = dummy;  
for (int i = 1; i < left; ++i) {  
    prev = prev->next;
```

```
struct ListNode * current = prev->next;
```

```
struct ListNode * next = NULL;
```

```
struct ListNode * tail = current;
```

```
for (int i = left; i <= right; i++) {
```

```
    struct ListNode * temp = current->next;
```

```
    current->next = next;
```

```
    next = current;
```

```
    current = temp;
```

```
    prev->next = next;
```

```
    tail = next = current;
```

```
struct ListNode * result = dummy->next;
```

```
free (dummy);
```

```
return result;
```

classmate
Date _____
Page _____

15/02/24

Week - 3 : Binary

#include <iostream>

#include <stdlib.h>

struct node {

int data;

struct node * left;

struct node * right;

};

struct node * newnode;

struct node * root;

root = newnode;

newnode->data = 1;

newnode->left = newnode;

newnode->right = newnode;

return root;

};

struct node * S;

if (x == 0)

return S;

if (x < 0)

return S;

else

return S;

x = 0;

return S;

};

void i

if (x < 0)

return S;

else

return S;

};

no

for (i = 0;

i < n;

i++)

};

15/02/24 Week - 3 : Binary search tree

#include <stdio.h>

#include <stdlib.h>

struct node {

int data;

struct node *left;

struct node *right;

};

struct node *newNode (int data) {

struct node *node = (struct node *) malloc

(sizeof (struct node)).

node->data = data;

node->left = node->right = NULL;

return node;

}

struct node *insert (struct node *root, int data)

{

if (root == NULL)

return newNode (data);

if (data <= root->data)

root->left = insert (root->left, data);

else

root->right = insert (root->right, data);

return root;

}

void inorder (struct node *temp) {

if (temp == NULL)

return;

inorder (temp->left);

printf ("%d", temp->data);

inorder (temp->right);

}

(root, left, right)

void preOrder (struct node *temp) {

if (temp == NULL)

return; // prints "1.d" if temp->data

preOrder (temp->left);

preOrder (temp->right);

} (left->right->root)

void postOrder (struct node *temp) {

if (temp == NULL)

return; // prints "1.d" if temp->data

postOrder (temp->left);

postOrder (temp->right);

printf ("1.d", temp->data);

} (left->right->root)

exit main ()

struct node *root = NULL;

int data, choice;

do {

printf ("Enter your choice: ");

scanf ("%d", &choice);

switch (choice) {

case 1: printf ("Enter the value to be inserted: ");

scanf ("%d", &data);

root = insert (root, data);

break;

case 2: printf ("Inorder traversal of binary tree:\n");

inOrder (root);

break;

case 3: printf ("Preorder traversal of binary tree:\n");

preOrder (root);

break;

case 4: printf ("Postorder traversal of binary tree:\n");

postOrder (root);

break;

case 5: printf ("Delete operation:\n");

scanf ("%d", &choice);

switch (choice) {

case 3: if(*n*if ("Preorder traversal of binary tree is \n"),
 preorder(*root*),
 break;

case 4: if(*n*if ("Postorder traversal of binary tree is \n"),
 postorder(*root*),
 break;

case 5: if(*n*if ("Exit"),
 break;

default: if(*n*if ("Invalid choice. Please try again"),
 {}
 white choice { = 5 })

 return 0;

}

Output

Main menu

1. insert
2. inorder traversal
3. postorder traversal
4. preorder traversal
5. exit

Enter the choice: 1

Enter the value to be inserted: 100

Enter the choice: 1

Enter the value to be inserted: 200

Enter the choice: 1

Enter the value to be inserted: 200

Enter the choice: 1

Enter the value to be inserted: 10

Enter the choice : 1

Enter the value to be inserted : 30

Enter the choice : 1

Enter the value to be inserted : 150

Enter the choice : 1

Enter the value to be inserted : 300

Enter the choice : 2

Inorder traversal : 10 20 30 100 150 200 300

Enter the choice : 3

Postorder traversal : 10 30 20 150 300 200 100

Enter the choice : 4

Preorder traversal : 100 20 10 30 200 150 300

split linked list in parts

typical of struct `listNode`:

`int getLen(listNode *head)`

`int n=0;`

`while(head)`

`{`

`n++;`

`head = head->next;`

`}`

`return n;`

`}`

`struct listNode ** splitListToParts(struct listNode *`

`*head, int k, int *returnSize`

`)`

`int n = getLen(head), elem, i, j;`

`*returnSize = k;`

`listNode ** list = (listNode **) malloc(k * sizeof(listNode)),`

`*t = head.`

`if(n > k)`

`{`

`for(i=0; i < k; i++)`

`{`

`elems = i < n ? n / k + 1 : n % k;`

`j = 0;`

`list[i] = head;`

`t = head.`

`while(j + i < elems)`

`{`

`t = head.`

`head = head->next;`

`}`

`t->next = NULL;`

~~N
1/2/24.~~

22/02/24

a) Write
BFS

core

{
for (i=0; i<n; i++)

list *i = head;

head = head->next;

list *i = next->next; NULL;

}

return list;

}

O/P

TIP: head = {1,2,3},

K = 5

Output: {{1},{2},{3},{4},{5}}

TIP: head = {1,2,3,4,5,6,7,8,9,10})

K = 3

Output: {{1,2,3,4},{5,6,7},{8,9,10}}

22/02/24

Week 9

- a) Write a program to traverse a graph using BFS

```
#include <stdio.h>
```

```
#define MAX_VERTICES 10
```

```
int n, i, j, visited[MAX_VERTICES],  
queue[MAX_VERTICES], front = 0, rear = 0;
```

```
void bfs (int v) {
```

```
    visited[v] = 1;
```

```
    queue[rear++] = v;
```

```
    while (front < rear) {
```

```
        int current = queue[front++];
```

```
        printf ("• %d ", current);
```

```
        for (i = 0; i < n; i++) {
```

```
            if (adj[current][i] == 1 && !visited[i]) {
```

```
                visited[i] = 1;
```

```
                queue[rear++] = i;
```

```
}
```

```
    } else main();
```

```
    exit (0);
```

```
    printf ("Enter the number of vertices");
```

```
    scanf ("%d", &n);
```

```
    for (i = 0; i < n; i++) {
```

```
        visited[i] = 0;
```

matrix
graph data in vector form

```
    printf ("Enter the starting vertex: ");
```

```
    for (i = 0; i < n; i++)
```

```
        for (j = 0; j < n; j++)
```

```
            scanf ("%d", &adj[i][j]);
```

```

        cout << "Enter the starting vertex: ";
        cin >> s;
        dfs(s);
        for (i = 0; i < n; i++) {
            if (!visited[i])
                cout << "InBFS is not feasible. Not all
                nodes are reachable in ";
        }
        return 0;
    }
}

```

Output:

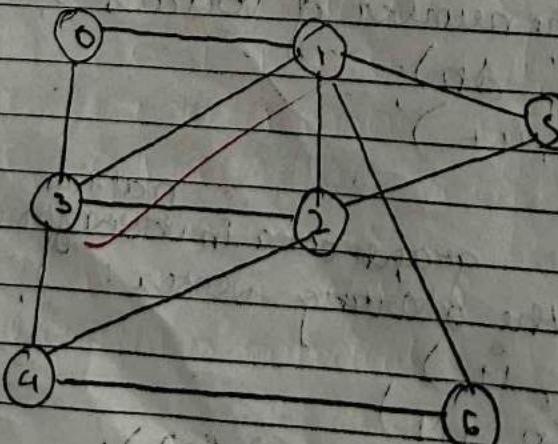
Enter the number of vertices: 7

Enter the adjacency matrix:

0	1	0	1	0	0	0
1	0	1	0	1	1	0
0	1	0	1	1	0	0
1	1	1	0	1	0	0
0	0	1	1	0	0	1
0	1	1	0	0	0	0
0	0	1	0	1	0	0

Enter the starting vertex: 2

2 1 3 4 5 0 6



(1) write a program as

#include <

#include <

#define r

void dfs (

int relax,

int vertex,

int i;

for (i :

if (a

dfs (

3

3

bool is cor

rMAX_VSR

bool vi

dfs (

int i,

for (i :

if (

3

3

return

)

3

return

(f) write a program to check whether the given graph is connected or not using DFS method

#include <stdio.h>

#include <stdbool.h>

#define MAX_VERTICES 10

void dfs (int graph[MAX_VERTICES][MAX_VERTICES],
int numVertices, bool visited[MAX_VERTICES],
int vertex) {

 visited[vertex] = true;

 int i;

 for (i=0; i<numVertices; ++i) {

 if (graph[vertex][i] == 1 && !visited[i]) {

 dfs(graph, numVertices, visited, i);

}

}

bool isconnected (int graph[MAX_VERTICES][MAX_VERTICES],
int numVertices) {

 bool visited[MAX_VERTICES] = {false};

 dfs (graph, numVertices, visited, 0);

 int i;

 for (i=0; i<numVertices; ++i) {

 if (!visited[i]) {

 return false;

}

}

return true;

}

```

    enter main () {
        cout << "Enter num vertices: ";
        cin >> num_vertices;
        cout << "Enter the number of vertices: ";
        cin >> n;
        cout << "Enter the adjacency matrix: ";
        for (int i = 0; i < num_vertices; ++i) {
            for (int j = 0; j < num_vertices; ++j) {
                cout << " " << adj[i][j];
            }
        }
        cout << endl;
        cout << "Enter graph (max vertices) [max vertices].";
        cin >> graph;
        cout << "Enter the adjacency matrix: [n]";
        for (int i = 0; i < n; ++i) {
            for (int j = 0; j < n; ++j) {
                cout << " " << graph[i][j];
            }
        }
        cout << endl;
        if (isConnected(graph, num_vertices)) {
            cout << "The graph is connected." << endl;
        } else {
            cout << "The graph is not connected." << endl;
        }
        return 0;
    }

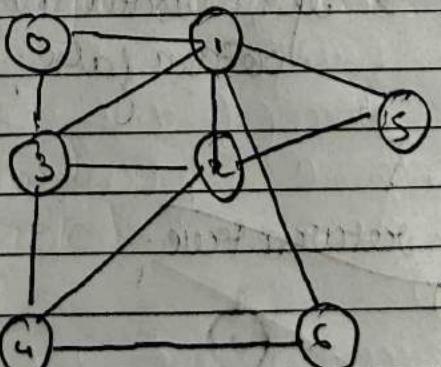
```

Output : *(With random seed 1234)*

Enter the number of vertices: ?

Enter the adjacency matrix:

0	1	0	1	0	0	0
1	0	1	1	0	1	1
0	1	0	1	1	1	0
1	1	1	0	1	0	0
0	0	1	1	0	0	1
0	1	1	0	0	0	0
0	1	0	0	1	0	0



The graph is connected

Rotate list

format : C:\USERS\PRASAD\POLYU

unit GetLength (struct listNode * head)

{
if (head == NULL)
return 0;

return 1 + GetLength (head->next);
}

struct listNode * rotateRight (struct listNode * head,
int k)

{
if (head == NULL || k == 0)
return head;

int length = GetLength (head);
if (length == 0)
return head;

if (length == 1 || k >= length)
return head;

for (int i = 0; i < k % length; i++)
{
struct listNode * p = head;

while (p->next != NULL)
p = p->next;

struct listNode * a = (struct listNode *) malloc
(sizeof (struct listNode));

a->val = p->next->val;
a->next = head;

head = a->next;
p->next = NULL;

return head;
}

Output
head = {1, 2, 3, 4, 5} k = 2

Output: {5, 1, 2, 3, 4}

head = {1, 2, 3, 4} k = 4

Output: {3, 4, 1, 2}

29/02/24 Week - 10 : Hashing

```
#include <stdio.h>
#include <stdlib.h>
#define MAX_EMPLOYEES 100
#define HASH_TABLE_SIZE 10
struct Employee {
    int key;
    char name[20];
}
```

3.

```
int hashFunction(int key);
void insertEmployee(struct Employee employees[], int hashTable[], struct Employee emp);
int hashTable[], struct Employee emp);
void displayHashTable(int hashTable[]);
```

int main () {

```
    struct Employee employees[MAX_EMPLOYEES];
    int hashTable [HASH_TABLE_SIZE] = {0};
    int n, i;
```

```
    printf ("Enter the number of employees : ");
    scanf ("%d", &n);
```

```
    printf ("Enter employee records : \n");
    for (i=0; i<n; i++) {
```

```
        printf ("Employee id: %d ", i+1);
        printf ("Enter 4-digit key : ");
```

```
        scanf ("%d", &employees[i].key);
        insertEmployee(employees, hashTable,
```

```
        employees[i].key);
    }
```

3

```
    printf ("In HashTable : \n");
    displayHashTable (hashTable);
```

```
    return 0;
}
```

exit hashFunction (int key) {

```
    return key % HASH_TABLE_SIZE;
```

void inser
ent hashT

int inde
while (n
index

3
hashTab

})
void ai
exit i,
for (i=

green
y (

fp
3 enc
fp

3
})

})

Output

Enter 5

Enter t

Employee

Enter

Emplo

Enter

Empl

Enter

Empl

Enter

Empl

Enter

Empl

Enter

Empl

Enter

```

void insertEmployee (struct Employee employee[],  

    int hashTable[], struct Employee emp) {  

    int index = hashFunction (emp.key);  

    while (hashTable[index] != 0) {  

        index = (index + 1) % HASH_TABLE_SIZE;  

    }  

    hashTable[index] = emp.key; // Inserting  

}  

void displayHashTable (int hashTable[]) {  

    int i;  

    for (i = 0; i < HASH_TABLE_SIZE; i++) {  

        if (hashTable[i] == -1) {  

            cout << "Empty" << endl;  

        } else {  

            cout << hashTable[i] << endl;  

        }
    }
}

```

Output:

Enter the number of employees : 10

Enter the employee records:

Employee 1 :

Enter 4-digit key 1234

Employee 2

Enter 4-digit key : 3456

Employee 3

Enter 4-digit key : 2678

Employee 4

Enter 4-digit key : 2789

Employee 5

Enter 4 digit key : 1254

Employee 6

Enter 4-digit key : 8970

Employee 9

Enter 4-digit key : 4878

Employee 8

Enter 4-digit key : 3241

Employee 9

Enter 4-digit key : 9895

Employee 10

Enter 4-digit key : 2456

Hash table

0 → 8970

1 → 3241

2 → 9895

3 → 2456

4 → 1234

5 → 2675

6 → 3436

7 → 1254

8 → 4528

9 → 72789

8970
3241
9895
2456
1234
2675
3436
1254
4528
72789

Hackerrank

```
#include <stdio.h>
```

```
#include <stdlib.h>
```

```
struct node
```

```
{
```

```
    int id;
```

```
    int depth;
```

```
    struct node *left, *right;
```

```
}
```

```
void inorder(struct node *tree);
```

```
{
```

```
    if (tree == NULL)
```

```
        return;
```

```
    inorder(tree->left);
```

```
    if (tree->id == 1)
```

```
        inorder(tree->right);
```

```
}
```

```
int main (void)
```

```
{
```

```
    int no_of_nodes, pi = 0;
```

```
    int l, s, max_depth, k;
```

 ~~struct node *lcmph = NULL;~~ ~~dcanf ("o1-d", &no_of_nodes);~~ ~~struct node *tree = (struct node *) malloc~~~~(no_of_nodes, sizeof (struct node));~~ ~~tree[0].depth = 1;~~ ~~while (pi < no_of_nodes)~~~~{~~ ~~tree[pi].id = pi + 1;~~ ~~dcanf ("o1-d-o1-d", &l, &s);~~ ~~if (l == -1)~~ ~~tree[pi].left = NULL;~~ ~~else~~~~{~~ ~~tree[pi].left = &tree[l-1];~~

$\text{tree}(i).right \rightarrow \text{depth} = \text{tree}(i).depth + 1$
 $\text{max_depth} = \text{tree}(i).depth \rightarrow \text{depth} + 2$

}
 }
 i++;

scanf("%d", &i);

while(i--) { input, loop starts from max_depth }

scanf("%d", &A[i]); // read binary number from user

i = l;

while(l <= max_depth) { loop - - - }

for(k=0; k< no. of nodes; k++) { loop - - - }

if (tree[k].depth == l)) { loop - - - }

temp = tree[k].left; move to

tree[k].left = tree[k].right;

tree[k].right = temp; move to

l = l + 1; (loop for left "bit")

union(u, tree); (union function)

freelist("in"); (free list function)

return 0;

(A, l, "in")

(l+1, 1)

l = l + 1

8/2
29/2

OUTPUT:

Input

3

2 3

-1 -1

-1 -1

2

1

1

Output

3 1 2

2 1 3

Input

17

2 3

4 5

6 -1

-1 7

8 9

10 11

12 13

-1 14

-1 -1

13 -1

8/12

10 12

-1 -1

9/12

-1 -1

-1 -1

-1 -1

-1 -1

-1 -1

2

Output

14 8 5 9 2 4 13 7 12 1 3
10 16 6 17 11 15 9 5 14 8 2
13 7 12 4 1 3 17 11 16 6 10 15