

Nama : Muhammad Iqbal Maulana
NIM : 1203230066
Kelas : IF-03-03

TUGAS ASD PRAKTIKUM ARRAY, POINTER DAN FUNGSI

Komponen Penilaian	Ya	Tidak
Soal 1 sesuai dengan output yang diinginkan	v	
Soal 2 sesuai dengan output yang diinginkan	v	
Bonus soal 1 dikerjakan	v	

1. Refan Judi

1.1 Screenshot Code

```
C refanJudi.c > ...
1  #include <stdio.h>
2  #include <ctype.h>
3  #include <stdlib.h>
4
5  int mapInput(char input)
6  {
7      switch (input)
8      {
9          case 'J':
10             return 11;
11
12          case 'Q':
13             return 12;
14
15          case 'K':
16             return 13;
17
18          default:
19             return -1;
20      }
21 }
22
23 void printOutput(int output)
24 {
25     switch (output)
26     {
27         case 11:
28             printf("J");
29             break;
30
31         case 12:
32             printf("Q");
33             break;
34
35         case 13:
36             printf("K");
37             break;
38
39         default:
40             printf("%d", output);
41             break;
42     }
43 }
44
```

```
1 // refanJudi.c
2 //
3 // Created by Refan on 10/10/2020.
4 //
5 // Copyright © 2020 Refan. All rights reserved.
6
7 #include <stdio.h>
8 #include <ctype.h>
9 #include <stdlib.h>
10
11 int mapInput(char input)
12 {
13     switch (input)
14     {
15         case 'J':
16             return 11;
17         case 'Q':
18             return 12;
19         case 'K':
20             return 13;
21         default:
22             return -1;
23     }
24 }
25
26 void printOutput(int output)
27 {
28     switch (output)
29     {
30         case 11:
31             printf("J");
32             break;
33         case 12:
34             printf("Q");
35             break;
36         case 13:
37             printf("K");
38             break;
39         default:
40             printf("%d", output);
41             break;
42     }
43 }
44
```

```

44
45 void swap(int *xp, int *yp, int swap_count, int count, int *numbers)
46 {
47     int temp = *xp;
48     *xp = *yp;
49     *yp = temp;
50
51     printf("Pertukaran %d: ", swap_count);
52     for (size_t i = 0; i < count; i++)
53     {
54         printOutput(*(numbers + i));
55         printf(" ");
56     }
57
58     printf("\n");
59 }
60
61 void sort(int *numbers, int count)
62 {
63     int swap_count = 0;
64     for (size_t i = 0; i < count - 1; i++)
65     {
66         int minIndex = i;
67         for (size_t j = i; j < count; j++)
68         {
69             if (numbers[j] < numbers[minIndex])
70             {
71                 minIndex = j;
72             }
73         }
74
75         if (minIndex != i)
76         {
77             swap_count++;
78             swap(numbers + minIndex, numbers + i, swap_count, count, numbers);
79         }
80     }
81
82     printf("%d\n", swap_count);
83 }
84

```

```

85 int main()
86 {
87     int count;
88     scanf("%d", &count);
89
90     int *numbers = (int *)malloc(count * sizeof(int));
91     if (numbers == NULL)
92     {
93         printf("Memory allocation failed.\n");
94         return 1;
95     }
96
97     for (int i = 0; i < count; i++)
98     {
99         char input;
100         scanf(" %c", &input);
101
102         int digit;
103         if (isdigit(input))
104         {
105             digit = input - '0';
106             if (digit < 0 || digit > 10)
107             {
108                 printf("Invalid input!\n");
109                 return 1;
110             }
111         }
112         else
113         {
114             digit = mapInput(input);
115             if (digit == -1)
116             {
117                 printf("Invalid input!\n");
118                 return 1;
119             }
120         }
121
122         numbers[i] = digit;
123     }
124
125     sort(numbers, count);
126
127     return 0;
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
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
620
621
622
623
624
625
626
627
628
629
630
631
632
633
634
635
636
637
638
639
640
641
642
643
644
645
646
647
648
649
650
651
652
653
654
655
656
657
658
659
660
661
662
663
664
665
666
667
668
669
670
671
672
673
674
675
676
677
678
679
680
681
682
683
684
685
686
687
688
689
690
691
692
693
694
695
696
697
698
699
700
701
702
703
704
705
706
707
708
709
710
711
712
713
714
715
716
717
718
719
720
721
722
723
724
725
726
727
728
729
730
731
732
733
734
735
736
737
738
739
740
741
742
743
744
745
746
747
748
749
750
751
752
753
754
755
756
757
758
759
760
761
762
763
764
765
766
767
768
769
770
771
772
773
774
775
776
777
778
779
780
781
782
783
784
785
786
787
788
789
790
791
792
793
794
795
796
797
798
799
800
801
802
803
804
805
806
807
808
809
810
811
812
813
814
815
816
817
818
819
820
821
822
823
824
825
826
827
828
829
830
831
832
833
834
835
836
837
838
839
840
841
842
843
844
845
846
847
848
849
850
851
852
853
854
855
856
857
858
859
860
861
862
863
864
865
866
867
868
869
870
871
872
873
874
875
876
877
878
879
880
881
882
883
884
885
886
887
888
889
890
891
892
893
894
895
896
897
898
899
900
901
902
903
904
905
906
907
908
909
910
911
912
913
914
915
916
917
918
919
920
921
922
923
924
925
926
927
928
929
930
931
932
933
934
935
936
937
938
939
940
941
942
943
944
945
946
947
948
949
950
951
952
953
954
955
956
957
958
959
960
961
962
963
964
965
966
967
968
969
970
971
972
973
974
975
976
977
978
979
980
981
982
983
984
985
986
987
988
989
990
991
992
993
994
995
996
997
998
999

```

1.2 Penjelasan Kode

```
int mapInput(char input)
{
    switch (input)
    {
        case 'J':
            return 11;

        case 'Q':
            return 12;

        case 'K':
            return 13;

        default:
            return -1;
    }
}
```

Baris kode diatas adalah fungsi untuk memetakan karakter kartu J, Q dan K ke dalam integer untuk memudahkan proses pengurutan.

```
void printOutput(int output)
{
    switch (output)
    {
        case 11:
            printf("J");
            break;

        case 12:
            printf("Q");
            break;

        case 13:
            printf("K");
            break;
    }
}
```

```

        default:
            printf("%d", output);
            break;
    }
}

```

Baris kode diatas adalah fungsi untuk memetakan kembali integer ke bentuk karakter kartu J, Q dan K yang mana dipakai ketika mencetak hasil pengurutan ke terminal.

```

void swap(int *xp, int *yp, int swap_count, int count, int *numbers)
{
    int temp = *xp;
    *xp = *yp;
    *yp = temp;

    printf("Pertukaran %d: ", swap_count);
    for (size_t i = 0; i < count; i++)
    {
        printOutput(*(numbers + i));
        printf(" ");
    }

    printf("\n");
}

void sort(int *numbers, int count)
{
    int swap_count = 0;
    for (size_t i = 0; i < count - 1; i++)
    {
        int minIndex = i;
        for (size_t j = i; j < count; j++)
        {
            if (numbers[j] < numbers[minIndex])
            {
                minIndex = j;
            }
        }
    }
}

```

```

    }

    if (minIndex != i)
    {
        swap_count++;
        swap(numbers + minIndex, numbers + i, swap_count, count,
numbers);
    }
}

printf("%d\n", swap_count);
}

```

Baris kode diatas ada 2 fungsi terkait yang melakukan tugas pengurutan kartu. Fungsi sorting yang dipakai adalah selection sort dengan modifikasi fungsi swap, yaitu penambahan perintah untuk mencetak pertukaran elemen yang terjadi ke terminal.

```

int main()
{
    int count;
    scanf("%d", &count);

    int *numbers = (int *)malloc(count * sizeof(int));
    if (numbers == NULL)
    {
        printf("Memory allocation failed.\n");
        return 1;
    }

    for (int i = 0; i < count; i++)
    {
        char input;
        scanf(" %c", &input);

        int digit;
        if (isdigit(input))
        {
            digit = input - '0';

```

```

        if (digit < 0 || digit > 10)
        {
            printf("Invalid input!\n");
            return 1;
        }
    }
    else
    {
        digit = mapInput(input);
        if (digit == -1)
        {
            printf("Invalid input!\n");
            return 1;
        }
    }

    numbers[i] = digit;
}

sort(numbers, count);

return 0;
}

```

Baris kode diatas adalah fungsi main dari program refan judi. Pada bagian awal fungsi main, user diminta memasukkan jumlah kartu yang akan diurutkan, kemudian dialokasikan array dinamis sesuai jumlah kartu dari input user. Selanjutnya, program akan membaca input masing-masing posisi kartu sebagai tipe data karakter.

Setelah membaca jenis kartu, program akan mengkonversi tipe data karakter menjadi integer, untuk kartu berupa gambar, maka konversi menggunakan fungsi mapInput() sebelumnya. Setelah konversi selesai fungsi sort() dipanggil untuk melakukan fungsi utama dari program untuk mengurutkan kartu.

1.3 Output

```
● → 3 ./refanJudi
6
8 K J Q 6 7 3 2
Pertukaran 1: 6 K J Q 8 7
Pertukaran 2: 6 7 J Q 8 K
Pertukaran 3: 6 7 8 Q J K
Pertukaran 4: 6 7 8 J Q K
4
● → 3 ./refanJudi
4
9 Q K J
Pertukaran 1: 9 J K Q
Pertukaran 2: 9 J Q K
2
● → 3 ./refanJudi
4
6 6 9 7
Pertukaran 1: 6 6 7 9
1
● → 3 ./refanJudi
5
3 2 8 7 4
Pertukaran 1: 2 3 8 7 4
Pertukaran 2: 2 3 4 7 8
2
○ → 3 █
```

2. Kobo Imaginary Chess

2.1 Screenshot Code

```
C koboImaginaryChess.c > koboImaginaryChess(int, int, int, int *)
1  #include <stdio.h>
2
3  void koboImaginaryChess(int i, int j, int size, int *chessBoard)
4  {
5      // possibility below only paired up if the index is on the same type, either odd or even
6      // for example, possibleY[1] only get paired with possibleX[1] or possibleX[3] (all index are odd)
7      // and, possibleY[0] only get paired with possibleX[0] or possibleX[2] (all index are even)
8      int possibleY[4] = {i + 2, i + 1, i - 2, i - 1};
9      int possibleX[4] = {j + 1, j + 2, j - 1, j - 2};
10
11     for (size_t k = 0; k < 4; k++)
12     {
13         if ((possibleY[k] < size) && (possibleY[k] >= 0))
14         {
15             // to match with index requirement, start the loop with the remainder of k / 2 (giving either 0 or 1)
16             // and increase the step by 2
17             for (size_t l = k % 2; l < 4; l += 2)
18             {
19                 if ((possibleX[l] < size) && (possibleX[l] >= 0))
20                 {
21                     // pointer reference on 2-dimensional array are flattened
22                     // meaning row 2 is located after the last element of row 1
23                     // thats why when accessing row > 1, we need to multiply the row by array size
24                     // after that to access element on that row, just add the column index
25                     *((chessBoard + possibleY[k] * size) + possibleX[l]) = 1;
26                 }
27             }
28         }
29     }
30 }
31
32 int main()
33 {
34     int size = 8;
35     int chessBoard[8][8] = {
36         {0, 0, 0, 0, 0, 0, 0, 0},
37         {0, 0, 0, 0, 0, 0, 0, 0},
38         {0, 0, 0, 0, 0, 0, 0, 0},
39         {0, 0, 0, 0, 0, 0, 0, 0},
40         {0, 0, 0, 0, 0, 0, 0, 0},
41         {0, 0, 0, 0, 0, 0, 0, 0},
42         {0, 0, 0, 0, 0, 0, 0, 0},
43         {0, 0, 0, 0, 0, 0, 0, 0},
44     };
45 }
```

```

31
32 int main()
33 {
34     int size = 8;
35     int chessBoard[8][8] = {
36         {0, 0, 0, 0, 0, 0, 0, 0},
37         {0, 0, 0, 0, 0, 0, 0, 0},
38         {0, 0, 0, 0, 0, 0, 0, 0},
39         {0, 0, 0, 0, 0, 0, 0, 0},
40         {0, 0, 0, 0, 0, 0, 0, 0},
41         {0, 0, 0, 0, 0, 0, 0, 0},
42         {0, 0, 0, 0, 0, 0, 0, 0},
43         {0, 0, 0, 0, 0, 0, 0, 0},
44     };
45
46     int i, j;
47     scanf("%d %d", &i, &j);
48
49     if (i < 0 || i >= size || j < 0 || j >= size)
50     {
51         printf("Input invalid!\n");
52         return 1;
53     }
54
55     koboImaginaryChess(i, j, size, (int *)chessBoard);
56
57     for (size_t k = 0; k < size; k++)
58     {
59         for (size_t l = 0; l < size; l++)
60         {
61             printf("%d ", chessBoard[k][l]);
62         }
63
64         printf("\n");
65     }
66
67     return 0;
68 }
69

```

2.2 Penjelasan Kode

```
void koboImaginaryChess(int i, int j, int size, int *chessBoard)
{
    // possibility below only paired up if the index is on the same
    // type, either odd or even
    // for example, possibleY[1] only get paired with possibleX[1] or
    // possibleX[3] (all index are odd)
    // and, possibleY[0] only get paired with possibleX[0] or
    // possibleX[2] (all index are even)
    int possibleY[4] = {i + 2, i + 1, i - 2, i - 1};
    int possibleX[4] = {j + 1, j + 2, j - 1, j - 2};

    for (size_t k = 0; k < 4; k++)
    {
        if ((possibleY[k] < size) && (possibleY[k] >= 0))
        {
            // to match with index requirement, start the loop with
            // the remainder of k / 2 (giving either 0 or 1)
            // and increase the step by 2
            for (size_t l = k % 2; l < 4; l += 2)
            {
                if ((possibleX[l] < size) && (possibleX[l] >= 0))
                {
                    // pointer reference on 2-dimensional array are
                    // flattened
                    // meaning row 2 is located after the last
                    // element of row 1
                    // thats why when accessing row > 1, we need to
                    // multiply the row by array size
                    // after that to access element on that row, just
                    // add the column index
                    *((chessBoard + possibleY[k] * size) +
                    possibleX[l]) = 1;
                }
            }
        }
    }
}
```

Baris kode diatas merupakan fungsi utama (bukan main function) yang memetakan seluruh kemungkinan letak bidak kuda pada papan catur berdasarkan posisi awalnya. Teknik yang dipakai adalah dengan memetakan 4 kemungkinan arah gerakan bidak kuda, yaitu ke atas, ke bawah, ke kiri dan ke kanan. Untuk setiap arah gerakan, masing-masing memiliki 2 kombinasi gerakan vertikal dan horizontal. Sebagai contoh untuk gerakan bidak keatas, bidak dapat bergerak antara 2 unit vertikal ditambah 1 unit horizontal ke kiri atau ditambah 1 unit ke kanan.

Fungsi ini memetakan semua kemungkinan koordinat final pergerakan bidak dalam array possibleX dan possibleY, dimana pasangan koordinat (x, y) didapat dari ganjil-genapnya indeks array tersebut. Sebagai contoh koordinat x indeks ke 1 (ganjil) memiliki kemungkinan pasangan koordinat y indeks ke 1 atau 3 (ganjil). Sementara, koordinat x indeks ke 2 (genap) memiliki kemungkinan pasangan koordinat y indeks ke 2 atau 4 (genap).

Semua kemungkinan koordinat (x, y) tersebut kemudian dievaluasi terhadap batas papan catur, yaitu 0 - 7 (inclusive). Koordinat yang melebihi batas ini tidak akan digunakan. Koordinat yang valid akan membalik nilai 0 pada papan catur menjadi 1.

```
int main()
{
    int size = 8;
    int chessBoard[8][8] = {
        {0, 0, 0, 0, 0, 0, 0, 0},
        {0, 0, 0, 0, 0, 0, 0, 0},
        {0, 0, 0, 0, 0, 0, 0, 0},
        {0, 0, 0, 0, 0, 0, 0, 0},
        {0, 0, 0, 0, 0, 0, 0, 0},
        {0, 0, 0, 0, 0, 0, 0, 0},
        {0, 0, 0, 0, 0, 0, 0, 0},
        {0, 0, 0, 0, 0, 0, 0, 0},
    };

    int i, j;
    scanf("%d %d", &i, &j);

    if (i < 0 || i >= size || j < 0 || j >= size)
    {
        printf("Input invalid!\n");
        return 1;
    }
}
```

```

koboImaginaryChess(i, j, size, (int *)chessBoard);

for (size_t k = 0; k < size; k++)
{
    for (size_t l = 0; l < size; l++)
    {
        printf("%d ", chessBoard[k][l]);
    }

    printf("\n");
}

return 0;
}

```

Baris kode diatas merupakan fungsi main dari program ini. Pada awal fungsi didefinisikan array 2D yang menampung 8x8 papan catur dengan inisialisasi nilai seluruhnya adalah 0. Kemudian program meminta input user sebagai koordinat awal bidak kuda, fungsi koboImaginaryChess() sebelumnya lalu dipanggil dan hasil update array oleh fungsi tersebut kemudian dicetak pada terminal.

2.3 Output

```
● → 3 ./koboImaginaryChess
2 2
0 1 0 1 0 0 0 0
1 0 0 0 1 0 0 0
0 0 0 0 0 0 0 0
1 0 0 0 1 0 0 0
0 1 0 1 0 0 0 0
0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0
● → 3 ./koboImaginaryChess
3 7
0 0 0 0 0 0 0 0
0 0 0 0 0 0 1 0
0 0 0 0 0 1 0 0
0 0 0 0 0 0 0 0
0 0 0 0 0 1 0 0
0 0 0 0 0 0 1 0
0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0
● → 3 ./koboImaginaryChess
5 7
0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0
0 0 0 0 0 0 1 0
0 0 0 0 0 1 0 0
0 0 0 0 0 0 0 0
0 0 0 0 0 1 0 0
0 0 0 0 0 0 1 0
● → 3 ./koboImaginaryChess
1 4
0 0 1 0 0 0 1 0
0 0 0 0 0 0 0 0
0 0 1 0 0 0 1 0
0 0 0 1 0 1 0 0
0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0
○ → 3 █
```