

Programming using C

week 04 practice session and coding

Name:**S.Jananejaipratha**

Department:**AIML-'B'**

Roll No.:**241501079**

Alia and Bob are playing a game called "Stone Game". There are n stones in total. Alia and Bob take turns to take stones. In each turn, a player can remove either one stone or two stones. The player who picks the last stone wins. They follow the "Sudoku-First" rule. Hence Alia is always the one to make the first move. You task is to find out whether Alia can win if both play the game optimally.

Input Format

First line contains T , which is the number of test cases. Each test case will contain n number of stones.

Output Format

Print "Yes" if Alia can win else print "No".

Constraints

$1 \leq T \leq 1000$

$1 \leq n \leq 10000$

Sample Input and Output

Input

```
2
1
4
```

Output

```
Yes
No
```

Assumptions (Initially requires 2 Ns)

```
1 2 3 4 5 6 7 8 9 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 99 100 101 102 103 104 105 106 107 108 109 110 111 112 113 114 115 116 117 118 119 120 121 122 123 124 125 126 127 128 129 130 131 132 133 134 135 136 137 138 139 140 141 142 143 144 145 146 147 148 149 150 151 152 153 154 155 156 157 158 159 160 161 162 163 164 165 166 167 168 169 170 171 172 173 174 175 176 177 178 179 180 181 182 183 184 185 186 187 188 189 190 191 192 193 194 195 196 197 198 199 200 201 202 203 204 205 206 207 208 209 210 211 212 213 214 215 216 217 218 219 220 221 222 223 224 225 226 227 228 229 230 231 232 233 234 235 236 237 238 239 240 241 242 243 244 245 246 247 248 249 250 251 252 253 254 255 256 257 258 259 260 261 262 263 264 265 266 267 268 269 270 271 272 273 274 275 276 277 278 279 280 281 282 283 284 285 286 287 288 289 290 291 292 293 294 295 296 297 298 299 300 301 302 303 304 305 306 307 308 309 310 311 312 313 314 315 316 317 318 319 320 321 322 323 324 325 326 327 328 329 330 331 332 333 334 335 336 337 338 339 340 341 342 343 344 345 346 347 348 349 350 351 352 353 354 355 356 357 358 359 360 361 362 363 364 365 366 367 368 369 370 371 372 373 374 375 376 377 378 379 380 381 382 383 384 385 386 387 388 389 390 391 392 393 394 395 396 397 398 399 400 401 402 403 404 405 406 407 408 409 410 411 412 413 414 415 416 417 418 419 420 421 422 423 424 425 426 427 428 429 430 431 432 433 434 435 436 437 438 439 440 441 442 443 444 445 446 447 448 449 450 451 452 453 454 455 456 457 458 459 460 461 462 463 464 465 466 467 468 469 470 471 472 473 474 475 476 477 478 479 480 481 482 483 484 485 486 487 488 489 490 491 492 493 494 495 496 497 498 499 500 501 502 503 504 505 506 507 508 509 510 511 512 513 514 515 516 517 518 519 520 521 522 523 524 525 526 527 528 529 530 531 532 533 534 535 536 537 538 539 540 541 542 543 544 545 546 547 548 549 550 551 552 553 554 555 556 557 558 559 560 561 562 563 564 565 566 567 568 569 570 571 572 573 574 575 576 577 578 579 580 581 582 583 584 585 586 587 588 589 590 591 592 593 594 595 596 597 598 599 600 601 602 603 604 605 606 607 608 609 610 611 612 613 614 615 616 617 618 619 620 621 622 623 624 625 626 627 628 629 630 631 632 633 634 635 636 637 638 639 640 641 642 643 644 645 646 647 648 649 650 651 652 653 654 655 656 657 658 659 660 661 662 663 664 665 666 667 668 669 670 671 672 673 674 675 676 677 678 679 680 681 682 683 684 685 686 687 688 689 690 691 692 693 694 695 696 697 698 699 700 701 702 703 704 705 706 707 708 709 710 711 712 713 714 715 716 717 718 719 720 721 722 723 724 725 726 727 728 729 730 731 732 733 734 735 736 737 738 739 740 741 742 743 744 745 746 747 748 749 750 751 752 753 754 755 756 757 758 759 760 761 762 763 764 765 766 767 768 769 770 771 772 773 774 775 776 777 778 779 780 781 782 783 784 785 786 787 788 789 790 791 792 793 794 795 796 797 798 799 800 801 802 803 804 805 806 807 808 809 810 811 812 813 814 815 816 817 818 819 820 821 822 823 824 825 826 827 828 829 830 831 832 833 834 835 836 837 838 839 840 841 842 843 844 845 846 847 848 849 850 851 852 853 854 855 856 857 858 859 860 861 862 863 864 865 866 867 868 869 870 871 872 873 874 875 876 877 878 879 880 881 882 883 884 885 886 887 888 889 890 891 892 893 894 895 896 897 898 899 900 901 902 903 904 905 906 907 908 909 910 911 912 913 914 915 916 917 918 919 920 921 922 923 924 925 926 927 928 929 930 931 932 933 934 935 936 937 938 939 940 941 942 943 944 945 946 947 948 949 950 951 952 953 954 955 956 957 958 959 960 961 962 963 964 965 966 967 968 969 970 971 972 973 974 975 976 977 978 979 980 981 982 983 984 985 986 987 988 989 990 991 992 993 994 995 996 997 998 999 1000
```

Input	Expected	Got
1	Yes	Yes
4	No	No

Result of tests: 100%

You are designing a system which prints out numbers with a unique right applied to each of them. The right is based on the number of closed paths in their pattern as given number.

The number of holes that each of the digits from 0 to 9 have are equal to the number of closed paths in the digit. Here values are:

- 1, 2, 3, 4, and 7 = 0 holes.
- 0, 6, 8, and 9 = 1 hole.
- 5 = 2 holes.

Given a number, you need to determine the sum of the number of holes for all of its digits. For example the number 879 has 3 holes.

Complete the program, it must read values as integers denoting the total number of holes in sum.

Constraints:

1 <= sum <= 100

Input Format: Four numbers testing.

There is one line of test containing a single integer sum. The value is positive.

Sample Input:

400

Sample Output:

3

Explanation:

Add distribution count for each digit: 4, 0 and 0. Return: 1 + 0 + 1 = 2.

Sample Case 1:

Sample Input:

1500

Sample Output:

4

Explanation:

Add distribution count for each digit: 1, 5, 0, 0. Return: 0 + 1 + 0 + 0 = 1.

Answer: (usually requires O(N))

```
def countHoles(n):  
    for i in range(10):  
        if i == 0:  
            holes[i] = 1  
        elif i == 1:  
            holes[i] = 0  
        elif i == 2:  
            holes[i] = 0  
        elif i == 3:  
            holes[i] = 0  
        elif i == 4:  
            holes[i] = 0  
        elif i == 5:  
            holes[i] = 2  
        elif i == 6:  
            holes[i] = 1  
        elif i == 7:  
            holes[i] = 0  
        elif i == 8:  
            holes[i] = 1  
        elif i == 9:  
            holes[i] = 1  
    return holes
```

	Input	Expected	Got	
✓	400	3	3	✓
✓	1500	4	4	✓

Passed all sample(s) ✓

The problem values have been a nice blend of coding and content. I'm pleased. There were people who gave a task to make a pair hand of items at the board table by distributing various items with different values. Most of us were up with a solution. But I'm not sure, category coding from 1-7. If the maximum price of the item printed is 10, then we can purchase any item only. We added the following example to prove to you.

(c) If we suppose the maximum price of all items is 14, then our possible coins of 27, 37, 43, 54, 60 is sufficient any item ranging from 57 till 91.

Now, finally, being a lower degree suggested that we could actually estimate the number of cases required and gave following distribution: 17, 21, 22. According to how my line can be put forward can give ranging from 17 to 22. Perhaps was important with both of these two, that is to help identify coming up with a minimum number of observations for any arbitrary new person (Mullard)

Input: Normal

Contains average 15 minutes for maximum price of the time period in Poland.

Output Format

There is a large literature describing the numerous aspects of development of value judgment.

Combinatorics

$$V = \omega^2 \frac{1}{2} \left(\frac{1}{\omega^2} + \frac{1}{\omega^2} \right)$$

Before the sample output for the mailing

Sample Input 1:

1

Sample Output 1:

1

Example Input 2:

Sample Output 2:

1

Captions

Page 10 of 10

(According to Roberts) [2, 17, 18, 19] were also discussed.

Refers per Month only: (1) (2) (3) (4) (5) (6) (7) (8) (9) (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) (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) (

Pay (and bonus) % Spill

(According to Subjects 27, 28, 29, 34, 36) read no distribution

Rates per Month: only \$1.52 (US) can be enough to pay for your savings from \$1 to \$1. Hence, even in 1 Month, a discount could also be \$1.52 (US). Hence, even in 12

Abstract *Abstractly* appears in 15.

```

1  private void bubbleSort() {
2      int n = arr.length;
3      while (n > 1) {
4          for (int i = 0; i < n - 1; i++) {
5              if (arr[i] > arr[i + 1]) {
6                  // swap
7                  int temp = arr[i];
8                  arr[i] = arr[i + 1];
9                  arr[i + 1] = temp;
10             }
11         }
12         n--;
13     }
14 }

```

	Input	Expected	Got	
✓	100	0	0	✓
✓	10	0	0	✓
✓	1000	0	0	✓
✓	10000	0	0	✓
✓	100000	0	0	✓

Status	Finished
Started	Monday, 20 December 2021, 5:53 PM
Completed	Friday, 25 November 2021, 9:23 AM
Duration	24 days 8 hours

Version 1

Created

Modified at 0:00

0 files

0 questions

A set of N numbers (separated by one space) is passed as input to the program. The program must identify the count of numbers where the number is odd number.

Input Format

The first line will contain the N numbers separated by one space.

Boundary Conditions

$1 \leq N \leq 50$

The value of the numbers can be from -99999999 to 99999999

Output Format

The count of numbers where the numbers are odd numbers.

Sample Input / Output 1:

Input

5 10 15 20 25 30 35 40 45 50

Output

5

Explanation

The numbers meeting the criteria are 5, 15, 25, 35, 45.

Answer: (possibly incorrect 0/0)

```
1 def countOddNumbers(N):
2     arr = list(map(int, input().split()))
3     count = 0
4     for i in range(0, N):
5         if arr[i] % 2 != 0:
6             count += 1
7     return count
8
9 if __name__ == '__main__':
10     N = int(input())
11     result = countOddNumbers(N)
12     print(result)
```

	Input	Expected	Got
✓	5 10 15 20 25 30 35 40 45 50	5	5 ✓

Passed all tests! ✓

Activate Windows
Go to Settings to activate Windows.

Correct 2

Failed

Number read at 4.00

Play

Submit

Given a number N , return true if and only if it is a confusing number, which satisfies the following condition:

We can rotate digits by 180 degrees to form new digits. When 0, 1, 6, 8, 9 are rotated 180 degrees, they become 0, 1, 9, 6, 0 respectively. When 2, 3, 4, 5 and 7 are rotated 180 degrees, they become invalid. A confusing number is a number that when rotated 180 degrees becomes a **different** number with each digit valid.

Example 1:

0 -> 0
Input: 0
Output: true
Explanation:
We get 0 after rotating 0, 0 is a valid number and 0 != 0.

Example 2:

69 -> 66
Input: 69
Output: true
Explanation:
We get 66 after rotating 69, 66 is a valid number and 66 != 69.

Example 3:

11 -> 11
Input: 11
Output: false
Explanation:
We get 11 after rotating 11, 11 is a valid number but the value remains the same, thus 11 is not a confusing number.

Note:

- $0 < N < 10^9$
- After the rotation we can ignore leading zeros, for example if after rotation we have 0009 then this number is considered as just 9.

Answer: (partially correct 0 %)

```
1 public boolean confusingNumber(int N) {
2     return false;
3 }
4
5 let x=N,y=1;
6 while(N>0 || y>0){
7     x=x*10+N%10;
8     y=y*10+N%10;
9     N=N/10;
10    y=y/10;
11 }
12 if(x==y){
13     return false;
14 }
15 return true;
16 }
```

	Input	Expected	Got	
✓	0	true	true	✓
✓	69	true	true	✓
✓	11	false	false	✓

Passed all tests! ✓

Activate Windows
Go to Settings to activate Windows.

Explanation 1

1. Cannot use item 1 because $d > 1$ and sum w has to be avoided at any time.
2. Hence, max total is achieved by sum $= d + d + 2$.

Sample Case 2:

Sample Input For Custom Testing

Sample Input 2

3
3

Sample Output 2

5

Explanation 2

$d = 2 + 3 = 5$, is the best case for maximum sum item.

Answer: (possibly wrong: 0 %)

```
1 //PseudoCode:
2 int main()
3 {
4     long long int n,x1,x2,w;
5     scanf("%lld %lld %lld",&n,&x1,&x2);
6     for(i = 1; i <= n; i++)
7     {
8         x1 = x1 + 1;
9         x2 = x2 + 1;
10    }
11    return 0;
12 }
13 //PseudoCode:
14 return 0;
15 }
16 }
17 }
18 }
```

	Input	Expected	Got
✓	3 3	5	5
✓	3 3	5	5
✓	3 3	5	5

Passed all tests ✓

Activate Windows
Go to Settings to activate Windows