**ZOHO ROUND 2 PROGRAMMING QUESTIONS**

**Program 1:**

**Sub Topic:** Arrays

**Sliding Window**

Given an array of numbers and a window of size k. Print the maximum of numbers inside the window for each step as the window moves from the beginning of the array.

**Input Format**

Input contains the array size , no of elements and the window size

**Output Format**

print the maximum of numbers

**Constraints**

1 <= size <= 1000

**Sample Input 1**

8

1 3 5 2 1 8 6 9

3

**Sample Output 1**

5 5 5 8 8 9

**Solution:**

#include<stdio.h>

#include<limits.h>

int main()

{

int size , arr[1000],ws,ctr,max = INT\_MIN,ctr1;

scanf("%d",&size);

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

scanf("%d ",&arr[ctr]);

scanf("%d",&ws);

for( ctr = 0 ; ctr<= size - ws ; ctr++)

{

for( ctr1 = 0 ; ctr1 < ws ; ctr1++)

{

if( max < arr[ctr+ctr1])

max = arr[ctr1+ctr];

}

printf("%d ",max);

max = INT\_MIN;

}

}

**Testcase 1:**

**Input:**

8

1 3 5 2 1 8 6 9

3

**Output:**

5 5 5 8 8 9

**Testcase 2:**

**Input:**

10

3 7 5 1 2 9 8 5 3 2

3

**Output:**

7 7 5 9 9 9 8 5

**Testcase 3**

**Input:**

10

1 2 3 4 5 6 7 8 9 10

5

**Output:**

5 6 7 8 9 10

**Testcase 4**

**Input:**

42

19 0 30 40 62 7 7 80 95 66 13 95 52 78 66 99 24 28 20 11 57 10 8 17 68 59 19 75 13 11 34 84 72 74 31 43 30 98 0 68 28 75

39

**Output:**

99 99 99 99

**Testcase 5:**

**Input:**

58

93 87 65 1 74 6 98 24 95 0 63 46 4 16 13 13 33 11 4 39 97 45 14 56 65 48 6 83 56 51 67 57 90 20 87 47 53 9 54 64 76 88 58 50 2 69 81 39 93 16 36 51 60 87 29 4 61 28

52

**Output:**

98 98 98 98 98

**Testcase 6**

**Input:**

30

19 59 82 52 47 22 88 31 9 22 66 89 70 18 21 1 4 14 69 41 89 25 91 96 23 57 16 21 67 55

22

**Output:**

89 91 96 96 96 96 96 96 96

**Testcase 7**

**Input:**

12

33 86 39 55 87 49 66 75 58 17 37 13

9

**Output:**

87 87 87 87

**Testcase 8**

**Input:**

168

893 639 144 556 370 698 29 659 333 944 872 843 304 883 543 324 509 665 706 219 526 407 42 719 774 398 20 410 500 189 563 154 922 487 333 168 88 735 845 168 6 377 825 683 482 579 644 177 703 660 390 3 543 651 778 368 849 155 345 407 926 981 942 237 465 847 369 710 150 724 936 513 21 388 475 63 374 358 142 820 574 227 837 754 427 60 714 26 309 839 955 660 246 637 938 161 859 744 909 479 91 91 455 125 432 2 14 368 251 973 743 869 861 47 734 135 307 907 995 452 297 209 689 75 171 141 633 488 544 512 903 43 168 188 538 862 935 250 698 254 570 103 829 596 242 110 299 852 870 391 979 293 830 873 989 58 974 867 482 896 135 498 904 938 964 526 770 626

131

**Output:**

995 995 995 995 995 995 995 995 995 995 995 995 995 995 995 995 995 995 995 995 995 995 995 995 995 995 995 995 995 995 995 995 995 995 995 995 995 995

**Testcase 9**

**Input:**

245

383 113 882 356 2 856 733 419 651 39 293 872 708 782 99 215 358 218 950 425 460 491 353 427 296 972 16 617 413 514 884 177 990 72 228 36 845 930 63 587 438 873 776 175 740 807 937 932 668 435 883 46 225 655 629 501 528 920 187 829 253 438 225 249 805 228 440 608 854 914 63 240 211 498 310 515 13 839 816 263 855 93 887 427 416 965 500 303 414 527 598 226 985 429 755 506 181 485 837 464 277 792 743 81 284 561 912 572 517 472 615 529 919 543 560 191 889 812 909 889 372 345 161 337 288 177 673 6 71 179 68 690 810 899 980 359 104 448 853 419 622 720 622 57 764 643 294 354 180 246 209 700 344 358 507 82 60 184 820 884 490 150 268 304 365 461 600 633 952 277 430 591 215 112 998 559 54 215 202 165 41 665 624 415 412 53 364 75 644 611 211 356 53 742 598 249 70 820 763 7 170 441 731 552 864 904 311 451 589 124 260 262 976 822 321 330 718 960 414 94 832 970 967 646 978 802 9 629 733 905 573 897 703 532 867 927 353 632 304 534 233 434 395 751 633

112

**Output:**

990 990 990 990 990 990 990 990 990 990 990 990 990 990 990 990 990 990 990 990 990 990 990 990 990 990 990 990 990 990 990 990 990 985 985 985 985 985 985 985 985 985 985 985 985 985 985 985 985 985 985 985 985 985 985 985 985 985 985 985 985 985 985 998 998 998 998 998 998 998 998 998 998 998 998 998 998 998 998 998 998 998 998 998 998 998 998 998 998 998 998 998 998 998 998 998 998 998 998 998 998 998 998 998 998 998 998 998 998 998 998 998 998 998 998 998 998 998 998 998 998 998 998 998 998 998 998 998 998 998 998 998 998 998

**Testcase 10**

**Input:**

572

981 240 269 662 146 400 892 920 59 544 147 942 245 146 532 773 800 327 645 206 266 450 256 826 383 450 712 328 128 948 30 826 753 321 749 375 864 340 131 237 211 863 97 396 824 317 405 979 451 651 150 373 820 603 319 668 887 769 32 570 352 2 185 171 574 601 832 839 343 324 21 30 260 414 708 629 23 471 43 520 434 649 381 88 530 244 22 727 476 316 200 890 775 857 821 567 914 320 990 598 71 776 91 516 144 975 772 686 960 510 391 292 414 826 411 254 966 485 613 75 128 821 761 79 213 563 291 684 327 550 662 989 0 32 149 855 646 457 584 105 613 324 746 134 62 55 816 104 695 422 532 191 899 74 995 954 411 846 299 533 140 731 562 372 243 948 195 79 863 532 714 214 190 15 217 295 362 575 908 125 141 136 35 659 155 469 453 354 504 420 205 296 761 599 136 130 375 525 276 627 897 545 392 132 593 137 343 126 611 153 502 598 961 617 33 241 274 693 889 145 92 675 185 798 143 490 33 904 364 281 634 579 934 598 658 236 352 456 779 692 184 159 384 291 738 842 23 638 479 105 325 980 472 757 70 38 13 340 172 125 875 418 555 656 702 700 290 926 459 958 996 392 465 96 608 299 448 4 621 357 307 291 290 5 562 489 45 120 716 382 747 250 731 350 984 522 776 387 263 857 832 306 891 571 687 228 199 670 532 509 99 380 533 42 744 924 749 910 183 326 834 214 148 222 220 711 985 944 972 332 360 245 321 684 176 677 88 955 453 782 357 447 545 795 317 374 223 160 878 278 369 573 569 38 504 82 791 347 205 330 954 733 30 795 701 188 987 101 527 508 865 264 219 657 702 635 59 207 926 234 887 64 918 192 129 116 983 54 51 20 718 34 931 696 529 771 909 130 276 107 97 954 204 775 807 658 668 268 776 757 95 386 479 188 493 149 32 778 953 225 840 545 874 510 654 83 81 692 112 38 650 369 744 650 207 993 40 188 121 773 491 144 472 647 751 786 341 820 880 718 286 867 623 114 986 158 955 432 645 567 102 192 302 965 955 861 648 622 998 966 934 566 541 480 234 305 725 978 135 454 49 234 357 118 327 902 182 715 674 38 919 731 135 16 544 79 762 147 504 623 921 392 27 777 573 781 597 209 124 797 595 475 225 149 744 295 808 71 709 629 268 564 869 545 628 918 344 840 278 900 254 284 282 850 743 867 547 128 744 695 57 25 633 946 178 191 597 817 314 460 705 219 201 276 432 321 924 58 486 15 622 881 116 844 408 404 475 15 199 431 901 890

286

**Output:**

996 996 996 996 996 996 996 996 996 996 996 996 996 996 996 996 996 996 996 996 996 996 996 996 996 996 996 996 996 996 996 996 996 996 996 996 996 996 996 996 996 996 996 996 996 996 996 996 996 996 996 996 996 996 996 996 996 996 996 996 996 996 996 996 996 996 996 996 996 996 996 996 996 996 996 996 996 996 996 996 996 996 996 996 996 996 996 996 996 996 996 996 996 996 996 996 996 996 996 996 996 996 996 996 996 996 996 996 996 996 996 996 996 996 996 996 996 996 996 996 996 996 996 996 996 996 996 996 996 996 996 996 996 996 996 996 996 996 996 996 996 996 996 996 996 996 996 996 996 996 996 996 996 996 996 996 996 996 996 996 996 996 996 996 996 996 996 996 996 996 996 996 996 996 996 996 996 996 996 996 996 996 996 998 998 998 998 998 998 998 998 998 998 998 998 998 998 998 998 998 998 998 998 998 998 998 998 998 998 998 998 998 998 998 998 998 998 998 998 998 998 998 998 998 998 998 998 998 998 998 998 998 998 998 998 998 998 998 998 998 998 998 998 998 998 998 998 998 998 998 998 998 998 998 998 998 998 998 998 998 998 998 998 998 998 998 998 998 998 998 998 998 998 998 998 998 998 998 998 998 998 998 998 998 998 998 998

**Testcase 11:**

**Input:**

572

981 240 269 662 146 400 892 920 59 544 147 942 245 146 532 773 800 327 645 206 266 450 256 826 383 450 712 328 128 948 30 826 753 321 749 375 864 340 131 237 211 863 97 396 824 317 405 979 451 651 150 373 820 603 319 668 887 769 32 570 352 2 185 171 574 601 832 839 343 324 21 30 260 414 708 629 23 471 43 520 434 649 381 88 530 244 22 727 476 316 200 890 775 857 821 567 914 320 990 598 71 776 91 516 144 975 772 686 960 510 391 292 414 826 411 254 966 485 613 75 128 821 761 79 213 563 291 684 327 550 662 989 0 32 149 855 646 457 584 105 613 324 746 134 62 55 816 104 695 422 532 191 899 74 995 954 411 846 299 533 140 731 562 372 243 948 195 79 863 532 714 214 190 15 217 295 362 575 908 125 141 136 35 659 155 469 453 354 504 420 205 296 761 599 136 130 375 525 276 627 897 545 392 132 593 137 343 126 611 153 502 598 961 617 33 241 274 693 889 145 92 675 185 798 143 490 33 904 364 281 634 579 934 598 658 236 352 456 779 692 184 159 384 291 738 842 23 638 479 105 325 980 472 757 70 38 13 340 172 125 875 418 555 656 702 700 290 926 459 958 996 392 465 96 608 299 448 4 621 357 307 291 290 5 562 489 45 120 716 382 747 250 731 350 984 522 776 387 263 857 832 306 891 571 687 228 199 670 532 509 99 380 533 42 744 924 749 910 183 326 834 214 148 222 220 711 985 944 972 332 360 245 321 684 176 677 88 955 453 782 357 447 545 795 317 374 223 160 878 278 369 573 569 38 504 82 791 347 205 330 954 733 30 795 701 188 987 101 527 508 865 264 219 657 702 635 59 207 926 234 887 64 918 192 129 116 983 54 51 20 718 34 931 696 529 771 909 130 276 107 97 954 204 775 807 658 668 268 776 757 95 386 479 188 493 149 32 778 953 225 840 545 874 510 654 83 81 692 112 38 650 369 744 650 207 993 40 188 121 773 491 144 472 647 751 786 341 820 880 718 286 867 623 114 986 158 955 432 645 567 102 192 302 965 955 861 648 622 998 966 934 566 541 480 234 305 725 978 135 454 49 234 357 118 327 902 182 715 674 38 919 731 135 16 544 79 762 147 504 623 921 392 27 777 573 781 597 209 124 797 595 475 225 149 744 295 808 71 709 629 268 564 869 545 628 918 344 840 278 900 254 284 282 850 743 867 547 128 744 695 57 25 633 946 178 191 597 817 314 460 705 219 201 276 432 321 924 58 486 15 622 881 116 844 408 404 475 15 199 431 901 890

286

**Output:**

996 996 996 996 996 996 996 996 996 996 996 996 996 996 996 996 996 996 996 996 996 996 996 996 996 996 996 996 996 996 996 996 996 996 996 996 996 996 996 996 996 996 996 996 996 996 996 996 996 996 996 996 996 996 996 996 996 996 996 996 996 996 996 996 996 996 996 996 996 996 996 996 996 996 996 996 996 996 996 996 996 996 996 996 996 996 996 996 996 996 996 996 996 996 996 996 996 996 996 996 996 996 996 996 996 996 996 996 996 996 996 996 996 996 996 996 996 996 996 996 996 996 996 996 996 996 996 996 996 996 996 996 996 996 996 996 996 996 996 996 996 996 996 996 996 996 996 996 996 996 996 996 996 996 996 996 996 996 996 996 996 996 996 996 996 996 996 996 996 996 996 996 996 996 996 996 996 996 996 996 996 996 996 998 998 998 998 998 998 998 998 998 998 998 998 998 998 998 998 998 998 998 998 998 998 998 998 998 998 998 998 998 998 998 998 998 998 998 998 998 998 998 998 998 998 998 998 998 998 998 998 998 998 998 998 998 998 998 998 998 998 998 998 998 998 998 998 998 998 998 998 998 998 998 998 998 998 998 998 998 998 998 998 998 998 998 998 998 998 998 998 998 998 998 998 998 998 998 998 998 998 998 998 998 998 998 998

**Testcase 12**

**Input:**

663

208 572 683 383 964 818 94 662 656 445 15 501 48 234 831 741 409 781 531 144 40 103 860 638 40 499 882 436 722 947 298 624 938 232 365 583 703 762 251 179 403 676 32 167 124 180 782 789 907 418 747 347 251 655 893 512 604 239 187 253 368 274 541 570 557 911 391 246 405 66 122 500 442 920 419 498 342 80 208 939 890 149 328 744 941 577 619 42 455 100 786 292 271 70 424 382 922 537 705 428 93 10 433 329 85 593 353 622 116 646 397 680 623 71 642 425 806 976 150 621 235 50 896 559 599 791 2 621 705 909 99 482 368 57 378 586 835 749 856 195 599 613 870 179 257 455 135 308 509 514 991 391 471 0 812 664 719 454 491 62 770 235 374 126 262 999 77 872 478 623 731 10 679 974 306 587 66 678 265 379 756 343 341 482 274 967 374 510 808 795 36 232 796 596 945 913 79 654 272 537 220 549 87 385 748 513 313 506 904 827 567 989 681 640 96 932 181 374 198 904 474 109 476 805 352 16 469 227 626 556 709 615 622 40 582 9 933 269 428 751 285 327 890 322 461 141 777 535 776 822 749 184 597 717 43 740 358 390 680 285 145 337 524 352 132 15 951 171 100 74 882 354 71 778 857 394 7 301 135 993 66 783 35 517 793 194 433 141 622 255 396 901 682 350 547 982 245 266 474 221 383 249 67 26 246 187 172 285 433 905 82 764 645 368 70 975 182 141 503 96 304 323 209 189 486 506 772 416 419 227 208 113 119 210 71 233 380 203 83 675 889 224 459 519 759 633 256 622 619 717 572 909 915 177 253 395 845 207 887 781 924 272 824 861 147 556 143 886 231 345 151 612 265 323 801 609 705 739 232 194 929 35 353 868 393 809 514 833 710 410 274 191 385 649 234 83 349 984 994 578 841 78 247 672 415 753 804 372 203 254 718 956 976 303 253 173 952 620 751 414 611 797 943 205 850 15 199 226 267 909 729 419 672 737 20 567 699 627 92 591 93 153 747 770 824 448 840 8 769 650 501 345 274 144 133 63 414 129 253 337 487 540 379 202 309 67 196 73 983 81 986 498 31 53 126 101 273 727 378 541 836 578 683 204 164 946 245 293 831 465 783 298 95 645 208 631 151 451 282 799 866 506 733 845 864 745 512 245 162 115 596 853 264 493 921 166 249 492 373 799 739 364 410 52 379 711 336 235 24 915 395 183 683 87 510 760 489 358 335 331 300 381 77 774 519 122 992 703 1 94 8 237 595 812 608 911 133 786 884 380 334 656 897 35 739 495 652 891 690 500 573 317 231 84 405 585 719 370 679 655 859 319 867 480 279 433 160 483 188 407 363 723 378 748 459 457 146 963 191 103 139 346 282 997 235 273 789 15 337 12 542 926 768 295 296 614 719 795 395 744 39 902 213 369 804 671 369 155 448 978 87 170 628 16 60 993 328 324 900 774 965 465 989 788 176 872 102 640 266 568 415 270 991 390 571 868 612 80 619 199 692 53 765

549

**Output:**

999 999 999 999 999 999 999 999 999 999 999 999 999 999 999 999 999 999 999 999 999 999 999 999 999 999 999 999 999 999 999 999 999 999 999 999 999 999 999 999 999 999 999 999 999 999 999 999 999 999 999 999 999 999 999 999 999 999 999 999 999 999 999 999 999 999 999 999 999 999 999 999 999 999 999 999 999 999 999 999 999 999 999 999 999 999 999 999 999 999 999 999 999 999 999 999 999 999 999 999 999 999 999 999 999 999 999 999 999 999 999 999 999 999 999

**Program 2**

**Sub topic:** Number Crunching

*Add 2 numbers in given Base*

Add two numbers in the given base without converting into base

**Input Format**

get two numbers and base

**Output Format**

display the sum

**Sample Input 1**

1010 11001 2

**Sample Output 1**

100011

**Sample Input 2**

123 13 4

**Sample Output 2**

202

**Solution:**

#include<stdio.h>

int main()

{

int num1 , num2 , base , carry , power, newnum , sum;

scanf("%d%d%d", &num1 , &num2 , &base);

power = 1;

carry = newnum = 0;

while( num1/power || num2/power )

{

sum = (((num1/power)%10) + ((num2/power)%10)) + carry;

newnum = (sum%base) \* power + newnum;

carry = sum / base;

power \*= 10;

}

if(carry)

newnum = carry \* power + newnum;

printf("%d" , newnum);

return 0;

}

**Testcase 1:**

**Input:**

1010 121001

**Output:**

100011

**Testcase 2:**

**Input:**

123 13 4

**Output**:

202

**Testcase 3:**

**Input:**

45236 123456 7

**Output:**

202025

**Testcase 4:**

**Input:**

122121 12012 3

**Output:**

211210

**Testcase 5:**

**Input:**

12345 64754 10

**Output:**

77099

**Testcase 6:**

**Input:**

3442 344210 5

**Output:**

403202

**Testcase 7:**

**Input:**

2323 2323 4

**Output:**

11312

**Testcase 8:**

**Input:**

63434 32674 8

**Output:**

116330

**Testcase 9:**

**Input:**

10110 111 2

**Output:**

11101

**Program 3**

**Sub Topic:** Arrays

***Alternate sort in unsorted array***

( no extra space)

**Sample Input 1**

9

23 7 8 30 18 12 6 28 16

**Sample Output 1**

23 7 18 12 16 28 8 30 6

**Solution**

#include<stdio.h>

void ASC\_BubbleSort(int\*arr , int N , int start)

{

int ind , flag , temp;

do

{

for(ind = start , flag = 0; ind < N- 2 ; ind+=2)

{

if(arr[ind] > arr[ind+2])

{

flag = 1;

temp = arr[ind];

arr[ind] = arr[ind+2];

arr[ind+2] = temp;

}

}

}while(flag == 1);

}

void DEC\_BubbleSort(int\*arr , int N , int start)

{

int ind , flag , temp;

do

{

for(ind = start , flag = 0; ind < N- 2 ; ind+=2)

{

if(arr[ind] < arr[ind+2])

{

flag = 1;

temp = arr[ind];

arr[ind] = arr[ind+2];

arr[ind+2] = temp;

}

}

}while(flag == 1);

}

int main()

{

int N, ind;

scanf("%d" ,&N);

int arr[N];

for(ind = 0 ; ind < N ; scanf("%d" , &arr[ind++]));

ASC\_BubbleSort(arr , N-1 , 1);

DEC\_BubbleSort(arr , N , 0);

for(ind = 0 ; ind < N ; printf("%d " , arr[ind++]));

}

**Testcase 1:**

**Input:**

9

23 7 8 30 18 12 6 28 16

**Output:**

23 7 18 12 16 28 8 30 6

**Testcase 2:**

**Input:**

12

47 85 2 8 92 35 92 89 20 58 587 24

**Output:**

587 8 92 35 92 58 47 85 20 89 2 24

**Testcase 3:**

**Input:**

7

1 2 3 4 5 6 7

**Output:**

7 2 5 4 3 6 1

**Testcase 4:**

**Input:**

7

10 20 30 40 50 60 70

**Output:**

70 20 50 40 30 60 10

**Testcase 5:**

**Input:**

8

487 7824 623 75 85 42 24 6745

**Output:**

623 42 487 75 85 7824 24 6745

**Testcase 6:**

**Input:**

4

4643 63 7842 745

**Output:**

7842 63 4643 745

**Testcase 7:**

**Input:**

5

4387 75348 2364 7856 23

**Output:**

4387 7856 2364 75348 23

**Testcase 8:**

**Input:**

8

347 743 632 8 25 854 452 8

**Output:**

632 8 452 743 347 854 25 8

**Testcase 9:**

**Input:**

6

375 784 834 98246 7542 87

**Output**

7542 784 834 98246 375 87

**Program 4**

**Sub Topic :** Arrays

***Array with threshold value***

 Given an array and a threshold value find the o/p

i/p {5,8,10,13,6,2}

threshold = 3

o/p count = 17

explanation:

Number          parts               counts

5                    {3,2}                2

8                     {3,3,2}             3

10                   {3,3,3,1}          4

13                   {3,3,3,3,1}       5

6                     {3,3}                2

2                     {2}                   1

**Input Format**

N - no of elements in an array

array of elements

threshold value

**Output Format**

display the count

**Sample Input 1**

6

5 8 10 13 6 2

3

**Sample Output 1**

17

**Solution:**

#include<stdio.h>

int main()

{

int N , threshold , ind, count = 0;

scanf("%d" , &N);

int arr[N];

for(ind = 0 ; ind < N ; scanf("%d" , &arr[ind++]));

scanf("%d" , &threshold);

for(ind = 0 ; ind < N ; ind++)

count += (arr[ind]/ threshold) + ((arr[ind] % threshold)==0 ? 0 : 1);

printf("%d " ,count);

return 0;

}

**Testcase 1:**

**Input:**

6

5 8 10 13 6 2

3

**Output:**

12

**Testcase 2:**

**Input:**

7

20 35 57 30 56 87 30

10

**Output:**

33

**Testcase 3:**

**Input:**

12

67 743 73 634 89 734 9 76 90 36 65 34

12

**Output:**

226

**Testcase 4:**

**Input:**

10

10 20 30 40 50 60 70 80 90 100

10

**Output:**

55

**Testcase 5:**

**Input:**

5

4387 78 89 87965 57

100

**Output:**

927

**Testcase 6:**

**Input:**

7

438 879 56 123 8421 853 892

400

**Output:**

35

**Testcase 7:**

**Input:**

7

54378 8953 426 85 8964 2 9000

1000

**Output:**

85

**Testcase 8:**

**Input:**

10

489 853 843 835 895 89 24 8953 853 8935

853

**Output:**

31

**Program 5**

**Sub Topic:** Arrays

***Convert number to words***

range is 0-999

**Sample Input 1**

234

**Sample Output 1**

two hundred and thirty four

**Sample Input 2**

200

**Sample Output 2**

two hundred only

**Solution:**

#include<stdio.h>

int main()

{

char one[21][10]={"zero" , "one" , "two" , "three" , "four" , "five" , "six" , "seven" ,

"eight" , "nine" ,"" , "eleven" , "twelve" , "thirteen" , "fourteen" , "fifteen" , "sixteen" ,

"seventeen" , "eighteen" , "nineteen"};

char tens[10][9]={"" , "ten","twenty","thirty","forty","fifty","sixty","seventy",

"eighty","ninety"};

int num;

scanf("%d" , &num);

if(num >=0 && num <= 99)

{

if( num >= 0 && num <=19) printf("%s" , one[num]);

else

{

printf("%s" , tens [ num / 10 ]);

if(num % 10 ) printf(" %s" , one[ num % 10]);

}

}

else if( num / 100)

{

printf("%s hundred" , one[ num / 100 ] );

if( num % 100 == 0) printf(" only");

else

{

int rem;

rem = num % 100;

if( rem >= 1 && rem <=19)

printf(" and %s" , one[rem]);

else

{

printf(" and %s" , tens [ rem / 10 ]);

if(num % 10 ) printf(" %s" , one[ rem % 10]);

}

}

}

return 0;

}

**Testcase1:**

**Input:**

234

**Output:**

two hundred and thirty four

**Testcase2:**

**Input:**

200

**Output:**

two hundred only

**Testcase3:**

**Input:**

203

**Output:**

two hundred and three

**Testcase4:**

**Input:**

0

**Output:**

zero

**Testcase5:**

**Input:**

545

**Output:**

five hundred and forty five

**Testcase6:**

**Input:**

999

**Output:**

nine hundred and ninety nine

**Testcase 7**

**Input:**

12

**Output:**

twelve

**Testcase 8**

**Input:**

34

**Output:**

thirty four

**Testcase 9**

**Input:**

550

**Output:**

five hundred and fifty

**Testcase 10**

**Input:**

1

**Output:**

one

**Program 6**

**Sub topic:** Number crunching

***cyclic number verification***

verify the given number is cyclic or not

**Input Format**

Num1 num2

**Constraints**

1<=range<=9999999999

**Sample Input 1**

12345 45123

**Sample Output 1**

Yes

**Sample Input 2**

12345 54123

**Sample Output 2**

No

**Solution:**

#include<stdio.h>

int main()

{

long long int num1 , num2 , power , nod , rotate;

scanf("%lld%lld",&num1 , &num2);

power = 1;

nod = 0;

while(num1 / power)

{

power \*= 10;

nod++;

}

power /= 10;

rotate = num1;

while(rotate != num2 && nod)

{

rotate = (rotate%10)\*power + (rotate/10);

nod--;

}

if(rotate == num2 && nod != 0 )

printf("Yes");

else

printf("No");

return 0;

}

**Testcase 1:**

**Input:**

12345 45123

**Output:**

Yes

**Testcase 2:**

**Input:**

12345 54123

**Output:**

No

**Testcase 3:**

**Input:**

123456789 678912345

**Output:**

Yes

**Testcase 4:**

**Input:**

123456789 679812345

**Output:**

No

**Testcase 5:**

**Input:**

1234678 6781234

**Output:**

Yes

**Testcase 6:**

**Input:**

1234678 6782134

**Output:**

No

**Testcase 7**

**Input:**

12345 12345

**Output:**

Yes

**Testcase 8**

**Input:**

12345 67892

**Output:**

No

**Testcase 9**

**Input:**

23456 62345

**Output:**

Yes

**Testcase 10**

**Input:**

2342432424 4234243242

**Output:**

Yes

**Testcase 11**

**Input:**

456 564

**Output:**

Yes

**Program 7**

**Sub Topic:** Strings

***Evaluate mathematical expression***

Check whether a given mathematical expression is valid.

**Sample Input 1**

(a+b)(c+d+e)

**Sample Output 1**

VALID

**Sample Input 2**

(a+b)(c+d)

**Sample Output 2**

VALID

**Sample Input 3**

(a+b))

**Sample Output 3**

INVALID

**Sample Input 4**

(ab+)

**Sample Output 4**

INVALID

**Solution:**

#include<stdio.h>

int strChr(char \*str , char ch)

{

int ind;

for(ind = 0 ; str[ind] ; ind++)

{

if(str[ind] == ch)

return 1;

}

return 0;

}

int main()

{

char str[100] , pop;

char close[5] = ")]" , open[5]="([" ,operators[6] ="+\*/-";

scanf("%s" , str);

char paran[100] , operand[100];

int top\_p , top\_o , ind , flag ;

top\_p = top\_o = -1;

for(ind = 0 ; str[ind] ; ind++)

{

if(strChr(open , str[ind]) == 1)

paran[++top\_p] = str[ind];

else if(strChr(close , str[ind]) == 1 )

{

if(top\_p == -1)

break;

else

{

pop = paran[top\_p];

flag = 0 ;

switch(pop)

{

case '(' : if(str[ind] != ')') flag = 1 ; break;

case '[' : if(str[ind] != ']') flag = 1 ; break;

case '{' : if(str[ind] != '}') flag = 1 ; break;

}

if(flag == 1) break;

else

paran[top\_p--] = 0;

}

}

else if( str[ind] >= 'a' && str[ind] <= 'z' )

operand[++top\_o] = str[ind];

else if( strChr(operators , str[ind]) == 1)

{

if( top\_o == -1) break;

else

{

operand[top\_o--] = 0;

if( !(str[ind+1] >= 'a' && str[ind+1] <= 'z'))

break;

else

{

ind++;

if(strChr(operators , str[ind+1]) == 1)

operand[++top\_o] = str[ind];

}

}

}

}

if(str[ind] == 0 && top\_o == -1 && top\_p == -1)

printf("VALID");

else

printf("INVALID");

return 0 ;

}

**Testcase 1**

**Input:**

(a+b)(c+d+e)

**Output:**

VALID

**Testcase 2**

**Input:**

(a+b)(c+d)

**Output:**

VALID

**Testcase 3**

**Input:**

(a+b))

**Output:**

INVALID

**Testcase 4**

**Input:**

(ab+)

**Output:**

INVALID

**Testcase 5**

**Input:**

(a+b+c)[(d+e)(f\*g)]

**Output:**

VALID

**Testcase 6**

**Input:**

(a+b))(c\*d)

**Output:**

INVALID

**Testcase 7**

**Input:**

(a+b/c-e)[a-b]

**Output:**

VALID

**Testcase 8**

**Input:**

(((a+b+c+d+e+f+g+h)))

**Output:**

VALID

**Testcase 9**

**Input:**

((ab+))

**Output**

INVALID

**Testcase 10**

**Input:**

(a+b)(c+d)(e+f)(a+b)(d+y+k+o)

**Output**

VALID

**Testcase 11**

**Input:**

(abdjcbc)

**Output**

INVALID

**Program 8**

**Subtopic:** Strings

***Find a Sub string***

    Find if a String2 is substring of String1. If it is, return the index of the first occurrence. else return -1.

**Sample Input 1**

thistest123string

123

**Sample Output 1**

8

**Solution:**

#include<stdio.h>

int main()

{

char str[200];

char substr[200];

scanf("%s %s", str , substr);

int ind , ind1 , len , sublen;

ind = 0 ;

for(len = 0 ; str[len] ; len++);

for(sublen = 0 ; substr[sublen] ; sublen++);

if(sublen > len)

printf("-1");

else

{

while(str[ind])

{

if( str[ind] == substr[0])

{

ind++;

ind1 = 1;

while(str[ind] && substr[ind1] && str[ind] == substr[ind1])

{

ind1++;

ind++;

}

if(substr[ind1] == 0 || str[ind] == 0)

break;

}

else

ind++;

}

if(substr[ind1] == 0)

printf("%d", ind - sublen);

else

printf("-1");

}

return 0;

}

**Testcase 1**

**Input:**

thistest123string

123

**Output:**

8

**Testcase 2**

**Input:**

haihellohaihello

ello

**Output:**

4

**Testcase 3**

**Input:**

wonder

ond

**Output:**

1

**Testcase 4**

**Input:**

haihello

how

**Output:**

-1

**Testcase 5**

**Input**

dsjkfsjdkvbsdkbv

sjdkvb

**Output:**

5

**Testcase 6**

**Input:**

jdfkbgvkjdfbvkjdfbvkjdfbjdfkbvkjdfb

vkjdfbvkjdfb

**Output:**

6

**Testcase 7**

**Input:**

dbfvdskjbvdskbvksdbvsdvbdshbvdksv

vbdshbvdksv

**Output:**

22

**Testcase 8**

**Input:**

dskbfvkjdgbvkfdASVCAHKSVHCsjgvdwiudytiWADC

ASVCAHKSVHC

**Output:**

15

**Testcase 9**

**Input:**

DKJVBDSKBVKDSVBKJDS

KBVKDSVBKJ

**Output:**

7

**Program 9**

**Sub Topic:** Strings

***Find the different pair***

print the pair which are mismatched in two strings

**Sample Input 1**

abcdefgh

abdfhjfb

**Sample Output 1**

c,d d,f e,h f,j g,f h,b

**Solution**

#include<stdio.h>

int main()

{

char str1[50];

char str2[50];

int ind;

scanf("%s %s" , str1 , str2);

for(ind = 0 ; str1[ind] ; ind++)

{

if(str1[ind] != str2[ind])

printf("%c,%c ", str1[ind] , str2[ind]);

}

return 0;

}

**Testcase 1:**

**Input:**

abcdefgh

abdfhjfb

**Output:**

c,d d,f e,h f,j g,f h,b

**Testcase 2:**

**Input:**

hjdhjjduk

hjdhjjdua

**Output:**

k,a

**Testcase 3**

**Input:**

dhjcdjcjdhvcjhdvchjdvs

dhjedjcjdhvfjhdvhhjdvs

**Output:**

c,e c,f c,h

**Testcase 4**

**Input:**

sakckjsgckjdscg

sakckjsgckjdsca

**Output:**

g,a

**Testcase 5**

**Input:**

dsjbvdjskbvj

dsjbddjgkbgj

**Output:**

v,d s,g v,g

**Testcase 6:**

**Input:**

sdjkcbdskjcb

sdjKcbDskJcb

**Output:**

k,K d,D j,J

**Testcase 7:**

**Input:**

askjcbgdskjcb

ashjcjjjskjcb

**Output:**

k,h b,j g,j d,j

**Testcase 8:**

**Input:**

dkjvbkjbvbjkf

dkjvfkjhvkjkf

**Output:**

b,f b,h b,k

**Program 10**

**Sub Topic:**Arrays

***First maximum and first minimum and so on***

*﻿*Given an array and arrange it with first maximum and first minimum and second maximum and second minimum and so on without using sorting and second array

**Sample Input 1**

15

5 15 10 25 55 35 75 45 95 50 70 40 60 90 3

**Sample Output 1**

95 3 90 5 75 10 70 15 60 25 55 35 50 40 45

**Solution:**

#include<stdio.h>

#include<malloc.h>

void swap(int\* , int,int);

void maxHeapify(int\* , int\* , int);

void minHeapify(int\* , int\* , int);

int main()

{

int ind , parent,left,right , n ;

//int arr[]={5,15,10,25,55,35,75,45,95,50,70,40,60,90,3};

int \*arr ;

scanf("%d" , &n);

arr = (int\*)malloc( n \* sizeof(int));

for(ind = 0 ; ind < n ; ind++)

scanf("%d" ,&arr[ind]);

int safe , count = 0;

safe = n ;

while(n!=0)

{

for(parent = n / 2 - 1;parent >=0 ; parent--)

{

if(count % 2 == 0)

maxHeapify(arr , &n , parent);

else

minHeapify(arr , &n , parent);

}

swap(arr , 0, --n);

count++;

}

for(ind = safe-1 ; ind >= 0 ; ind--)

printf("%d " , arr[ind]);

return 0;

}

void swap(int\*arr , int pos1,int pos2)

{

int temp;

temp = arr[pos1];

arr[pos1]= arr[pos2];

arr[pos2]= temp;

}

void maxHeapify(int\*arr , int \*size , int parent)

{

int left,right;

left= parent \* 2+1;

right = left + 1;

// no child

if(left >= \*size) return;

// 1child

if(right >= \*size)

{

if(arr[parent] < arr[left])

{

swap(arr , left,parent);

maxHeapify(arr , size , left);

}

return ;

}

// 2 child

if(arr[parent] < arr[left] || arr[parent] < arr[right])

{

if(arr[left] > arr[right])

{

swap(arr, left,parent);

maxHeapify(arr , size , left);

}

else

{

swap(arr , right,parent);

maxHeapify(arr , size , right);

}

}

}

void minHeapify(int\*arr , int \*size , int parent)

{

int left,right;

left= parent \* 2+1;

right = left + 1;

// no child

if(left >= \*size) return;

// 1child

if(right >= \*size)

{

if(arr[parent] > arr[left])

{

swap(arr , left,parent);

maxHeapify(arr , size , left);

}

return ;

}

// 2 child

if(arr[parent] > arr[left] || arr[parent] > arr[right])

{

if(arr[left] < arr[right])

{

swap(arr, left,parent);

minHeapify(arr , size , left);

}

else

{

swap(arr , right,parent);

minHeapify(arr , size , right);

}

}

}

**Testcase 1:**

**Input:**

15

5 15 10 25 55 35 75 45 95 50 70 40 60 90 3

**Output:**

95 3 90 5 75 10 70 15 60 25 55 35 50 40 45

**Testcase 2**

**Input:**

7

7 9 3 6 8 2 1

**Output:**

9 1 8 2 7 3 6

**Testcase 3**

**Input:**

7

467 742 8 32 8 489 26

**Output:**

742 8 489 8 467 26 32

**Testcase 4:**

**Input:**

8

200 4 84 82 62 8 38 643

**Output:**

643 4 200 8 84 38 82 62

**Testcase 5:**

**Input:**

7

486 73 8942 2742 831913 947 1

**Output:**

831913 1 8942 73 2742 486 947

**Testcase 6:**

**Input:**

8

47 743 5 853 835 735 53 88

**Output:**

853 5 835 47 743 53 735 88

**Testcase 7:**

**Input:**

13

44 556 235 56 546 6564 45 35 2 678 4367 75 7783

**Output:**

7783 2 6564 35 4367 44 678 45 556 56 546 75 235

**Testcase 8:**

**Input:**

10

45 454 437 7 98 74 6724 89 674 7

**Output:**

6724 7 674 7 454 45 437 74 98 89

**Testcase 9:**

**Input:**

7

378 75 85 85 623 9 89

**Output:**

7

378 75 85 85 623 9 89

**Program 11**

**Sub Topic:** Algorithms

Leap Year

find whether the given year is leap year or not

**Sample Input 1**

1990

**Sample Output 1**

Non-leap

**Sample Input 2**

2000

**Sample Output 2**

Leap

**Solution:**

#include<stdio.h>

int main()

{

int year;

scanf("%d" , &year);

if(year % 4 == 0 && year % 100 != 0 || year % 400 == 0)

printf("Leap");

else

printf("Non-leap");

return 0;

}

**Testcase 1:**

**Input:**

1990

**Output:**

Non\_Leap

**Testcase 2:**

**Input:**

2000

**Output:**

Leap

**Testcase 3:**

**Input:**

1987

**Output:**

Non-Leap

**Testcase 4:**

**Input:**

2020

**Output:**

Leap

**Testcase 5**

**Input:**

2400

**Output:**

Leap

**Testcase 6**

**Input:**

2345

**Output:**

Non-Leap

**Testcase 7**

**Input:**

3400

**Output:**

Non-Leap

**Testcase 8**

**Input:**

3600

**Output:**

Leap

**Testcase 9:**

**Input:**

2478

**Output:**

Non-Leap

**Testcase 10:**

**Input:**

4560

**Output:**

Leap

**Testcase 11:**

**Input:**

6747

**Output:**

Non-Leap

**Testcase 12:**

**Input:**

6727

**Output:**

Non-Leap

**Program 12**

**Sub topic:** Dynamic Memory

***Merge two sorted arrays without duplication***

Output is a merged array without duplicates

**Input Format**

N1 - no of elements in array 1

array elements for array 1

N2 - no of elements in array 2

array elements for array2

**Output Format**

display the merged array

**Sample Input 1**

5

1 2 3 6 9

4

2 4 5 10

**Sample Output 1**

1 2 3 4 5 6 9 10

**Solution**

#include<stdio.h>

#include<malloc.h>

int main()

{

int N1 , N2, ind, ind1;

scanf("%d", &N1);

int arr[N1];

for(ind = 0 ; ind < N1 ; scanf("%d" , &arr[ind++]));

scanf("%d" ,&N2);

int arr1[N2];

int \*newarr , newind ;

newarr = (int\*)calloc(N1+N2 , sizeof(int));

for(ind = 0 ; ind < N2 ; scanf("%d" , &arr1[ind++]));

ind = ind1 = newind = 0 ;

while(ind < N1 && ind1 < N2)

{

if(arr[ind] == arr1[ind1])

{

newarr[newind++] = arr[ind];

ind++;

ind1++;

}

else if( arr[ind] < arr1[ind1])

newarr[newind++] = arr[ind++];

else if(arr[ind] > arr1[ind1])

newarr[newind++] = arr1[ind1++];

}

while(ind == N1 && ind1 < N2)

newarr[newind++] = arr1[ind1++];

while(ind1 == N2 && ind < N1)

newarr[newind++] = arr[ind++];

for(ind = 0 ; ind < newind ; ind++)

printf("%d ", newarr[ind]);

return 0;

}

**Testcase 1**

**Input**

5

1 2 3 6 9

4

2 4 5 10

**Output:**

1 2 3 4 5 6 9 10

**Testcase 2**

**Input:**

7

4 7 8 10 12 30 35

9

1 3 4 5 7 8 11 13 22

**Output:**

1 3 4 5 7 8 10 11 12 13 22 30 35

**Testcase 3**

**Input:**

10

1 2 3 4 5 6 7 8 9 10

10

1 2 3 11 12 13 14 15 16 17

**Output:**

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

**Testcase 4**

**Input:**

5

1 2 3 4 5

5

1 2 3 4 5

**Output:**

1 2 3 4 5

**Testcase 5**

**Input:**

8

20 22 34 56 78 90 120 200

5

35 80 99 125 250

**Output:**

20 22 34 35 56 78 80 90 99 120 125 200 250

**Testcase 6**

**Input:**

7

22 33 44 55 66 77 88

10

11 20 22 30 33 40 44 50 55 60

**Output:**

11 20 22 30 33 40 44 50 55 60 66 77 88

**Testcase 7**

**Input:**

10

1 2 3 4 5 6 7 8 9 10

15

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

**Output:**

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

**Testcase 8**

**Input:**

7

578 689 789 900 1200 1345 1567

5

689 789 900 2000 2500

**Output:**

578 689 789 900 1200 1345 1567 2000 2500

**Program 13**

**Sub Topic:** Number Crunching

***New Number System***

 Form a number system with only 3 and 4. Find the nth number of the number system.

Eg.) The numbers are: 3, 4, 33, 34, 43, 44, 333, 334, 343, 344, 433, 434, 443, 444, 3333, 3334, 3343, 3344, 3433, 3434, 3443, 3444 ….

**Constraints**

1<=N<=10000000

**Sample Input 1**

10

**Sample Output 1**

344

**Sample Input 2**

6743

**Sample Output 2**

434334344333

**Solution:**

#include<stdio.h>

int main()

{

long long int n , start , end , count , ctr;

scanf("%lld" , &n);

long long int arr[n];

start = 0 ;

end = 1;

count = 2;

arr[0] = 3;

arr[1] = 4;

while(count <= n )

{

for(ctr = start ; ctr <= end && count <= n ; ctr++)

{

arr[count++] = arr[ctr] \* 10LL + 3;

arr[count++] = arr[ctr] \* 10LL + 4;

}

start = end + 1;

end = count - 1;

}

printf("%lld" , arr[n-1]);

return 0;

}

**Testcase 1:**

**Input:**

10

**Output:**

344

**Testcase 2:**

**Input:**

6743

**Output:**

434334344333

**Testcase 3**

**Input:**

1000

**Output:**

444434334

**Testcase 4**

**Input:**

567

**Output:**

333444333

**Testcase 5**

**Input:**

33

**Output:**

33343

**Testcase 6**

**Input:**

3456

**Output:**

43443333334

**Testcase 7**

**Input:**

46474

**Output:**

344343443334344

**Testcase 8**

**Input:**

300

**Output:**

33434434

**Testcase 9**

**Input:**

482874

**Output:**

443434444333444344

**Testcase 10**

**Input:**

648438

**Output:**

3344443343344443444

**Testcase 11**

**Input:**

342

**Output:**

34343444

**Program 14**

**Sub Topic:** Array

***Number and its occurrence***

*﻿*Given a array with n elements print the number of occurrences of that number each number in that array. The order of number doesn’t matter. You can reorder the elements.

**Input Format**

N - no of elements

array of elements

**Output Format**

display number followed by counts

**Sample Input 1**

10

4 7 18 16 14 16 7 13 10 2

**Sample Output 1**

4-1

7-2

18-1

16-2

14-1

13-1

10-1

2-1

**Solution:**

#include<stdio.h>

#include<malloc.h>

int main()

{

int N , ind , num , counter , flag , ind1;

scanf("%d" , &N);

int \*arr , \*count;

arr= (int\*)calloc(N , sizeof(int));

count= (int\*)calloc(N , sizeof(int));

counter = 0;

for(ind = 0 ; ind < N ; ind++)

{

scanf("%d" ,&num);

for(ind1 = 0, flag = 0 ; ind1 < counter ; ind1++)

{

if(arr[ind1] == num)

{

count[ind1]++;

flag = 1;

}

}

if(flag == 0)

{

arr[counter] = num;

count[counter]++;

counter++;

}

}

for(ind = 0 ; ind < counter ; ind++)

printf("%d-%d\n" , arr[ind] , count[ind]);

return 0;

}

**Testcase 1:**

**Input:**

10

4 7 18 16 14 16 7 13 10 2

**Output:**

4-1

7-2

18-1

16-2

14-1

13-1

10-1

2-1

**Testcase 2:**

**Input:**

50

184 87 178 116 194 136 187 93 50 22 163 28 91 60 164 127 141 27 173 137 12 169 168 30 183 131 63 124 68 136 130 3 23 59 70 168 194 57 12 43 30 174 22 120 185 138 199 125 116 171

**Output:**

184-1

87-1

178-1

116-2

194-2

136-2

187-1

93-1

50-1

22-2

163-1

28-1

91-1

60-1

164-1

127-1

141-1

27-1

173-1

137-1

12-2

169-1

168-2

30-2

183-1

131-1

63-1

124-1

68-1

130-1

3-1

23-1

59-1

70-1

57-1

43-1

174-1

120-1

185-1

138-1

199-1

125-1

171-1

**Testcase 3:**

**Input:**

150

184 87 178 116 194 136 187 93 50 22 163 28 91 60 164 127 141 27 173 137 12 169 168 30 183 131 63 124 68 136 130 3 23 59 70 168 194 57 12 43 30 174 22 120 185 138 199 125 116 171 14 127 92 181 157 74 63 171 197 82 106 126 85 128 137 106 47 130 114 58 125 96 183 146 15 168 35 165 44 151 88 9 77 179 189 185 4 52 155 200 133 61 77 169 140 13 27 187 95 140 196 171 35 179 68 2 98 103 118 93 53 157 102 81 87 42 66 90 45 20 41 130 32 118 98 172 82 76 110 128 168 57 98 154 187 166 107 84 20 25 129 72 133 30 104 20 71 169 109 116

**Output:**

184-1

87-2

178-1

116-3

194-2

136-2

187-3

93-2

50-1

22-2

163-1

28-1

91-1

60-1

164-1

127-2

141-1

27-2

173-1

137-2

12-2

169-3

168-4

30-3

183-2

131-1

63-2

124-1

68-2

130-3

3-1

23-1

59-1

70-1

57-2

43-1

174-1

120-1

185-2

138-1

199-1

125-2

171-3

14-1

92-1

181-1

157-2

74-1

197-1

82-2

106-2

126-1

85-1

128-2

47-1

114-1

58-1

96-1

146-1

15-1

35-2

165-1

44-1

151-1

88-1

9-1

77-2

179-2

189-1

4-1

52-1

155-1

200-1

133-2

61-1

140-2

13-1

95-1

196-1

2-1

98-3

103-1

118-2

53-1

102-1

81-1

42-1

66-1

90-1

45-1

20-3

41-1

32-1

172-1

76-1

110-1

154-1

166-1

107-1

84-1

25-1

129-1

72-1

104-1

71-1

109-1

**Testcase 4:**

**Input:**

75

1384 887 778 916 1794 336 1387 493 650 1422 363 28 691 60 1764 1927 541 1427 1173 1737 1212 1369 568 430 1783 1531 863 1124 68 1136 1930 1803 23 1059 1070 168 1394 457 1012 43 230 1374 422 920 1785 538 1199 325 316 371 414 1527 92 981 1957 1874 863 1171 997 1282 306 926 1085 328 337 506 847 1730 1314 1858 125 1896 1583 546 815

**Output:**

1384-1

887-1

778-1

916-1

1794-1

336-1

1387-1

493-1

650-1

1422-1

363-1

28-1

691-1

60-1

1764-1

1927-1

541-1

1427-1

1173-1

1737-1

1212-1

1369-1

568-1

430-1

1783-1

1531-1

863-2

1124-1

68-1

1136-1

1930-1

1803-1

23-1

1059-1

1070-1

168-1

1394-1

457-1

1012-1

43-1

230-1

1374-1

422-1

920-1

1785-1

538-1

1199-1

325-1

316-1

371-1

414-1

1527-1

92-1

981-1

1957-1

1874-1

1171-1

997-1

1282-1

306-1

926-1

1085-1

328-1

337-1

506-1

847-1

1730-1

1314-1

1858-1

125-1

1896-1

1583-1

546-1

815-1

**Testcase 5:**

**Input:**

121

4 7 8 6 4 6 7 3 10 2 3 8 1 10 4 7 1 7 3 7 2 9 8 10 3 1 3 4 8 6 10 3 3 9 10 8 4 7 2 3 10 4 2 10 5 8 9 5 6 1 4 7 2 1 7 4 3 1 7 2 6 6 5 8 7 6 7 10 4 8 5 6 3 6 5 8 5 5 4 1 8 9 7 9 9 5 4 2 5 10 3 1 7 9 10 3 7 7 5 10 6 1 5 9 8 2 8 3 8 3 3 7 2 1 7 2 6 10 5 10 1

**Output:**

4-11

7-17

8-13

6-11

3-15

10-13

2-10

1-11

9-8

5-12

**Testcase 6:**

**Input:**

500

4 7 8 6 4 6 7 3 10 2 3 8 1 10 4 7 1 7 3 7 2 9 8 10 3 1 3 4 8 6 10 3 3 9 10 8 4 7 2 3 10 4 2 10 5 8 9 5 6 1 4 7 2 1 7 4 3 1 7 2 6 6 5 8 7 6 7 10 4 8 5 6 3 6 5 8 5 5 4 1 8 9 7 9 9 5 4 2 5 10 3 1 7 9 10 3 7 7 5 10 6 1 5 9 8 2 8 3 8 3 3 7 2 1 7 2 6 10 5 10 1 10 2 8 8 2 2 6 10 8 8 7 8 4 7 6 7 4 10 5 9 2 3 10 4 10 1 9 9 6 1 10 7 4 9 6 7 2 2 6 10 9 5 9 2 1 4 1 5 5 5 5 8 7 4 2 8 6 10 7 3 2 8 9 6 8 5 2 9 6 10 8 6 4 9 9 4 2 9 10 7 5 4 4 4 9 7 1 5 9 9 9 10 8 8 7 5 4 1 4 1 10 3 6 5 1 6 10 5 7 10 3 3 5 8 8 6 5 9 2 3 9 10 4 7 9 1 3 2 1 6 2 2 1 9 6 1 7 5 7 3 6 9 7 3 9 5 8 3 5 1 7 3 10 10 1 9 2 4 2 2 1 4 5 1 4 10 2 10 7 10 4 4 9 1 6 7 7 5 1 1 5 7 3 7 8 6 7 10 9 8 3 9 3 10 10 7 1 3 8 7 2 4 3 2 6 10 10 2 5 10 2 1 8 6 9 8 1 5 9 1 5 3 10 7 2 1 5 3 3 3 1 6 6 3 10 1 3 9 4 9 1 5 1 10 2 10 7 3 6 5 5 10 10 4 7 1 6 1 3 10 5 4 6 2 8 5 4 2 5 7 10 5 3 3 7 5 2 3 9 9 10 3 9 9 9 7 9 4 9 4 4 4 9 1 5 8 7 9 10 1 7 9 8 10 1 4 4 4 8 4 3 7 6 3 7 6 9 8 10 7 1 5 2 1 5 9 8 1 9 7 3 5 8 10 4 10 3 9 4 1 2 8 9 10 2 6 5 10 3 6 8 5 10 10 5 6 10 4 6 8 1 9 2 10 10 8 9 3 6 4 5 10 1 3 1 2 10 7 3

**Output:**

4-47

7-54

8-44

6-42

3-51

10-62

2-43

1-51

9-54

5-52

**Testcase 7:**

**Input:**

1500

4 7 8 6 4 6 7 3 10 2 3 8 1 10 4 7 1 7 3 7 2 9 8 10 3 1 3 4 8 6 10 3 3 9 10 8 4 7 2 3 10 4 2 10 5 8 9 5 6 1 4 7 2 1 7 4 3 1 7 2 6 6 5 8 7 6 7 10 4 8 5 6 3 6 5 8 5 5 4 1 8 9 7 9 9 5 4 2 5 10 3 1 7 9 10 3 7 7 5 10 6 1 5 9 8 2 8 3 8 3 3 7 2 1 7 2 6 10 5 10 1 10 2 8 8 2 2 6 10 8 8 7 8 4 7 6 7 4 10 5 9 2 3 10 4 10 1 9 9 6 1 10 7 4 9 6 7 2 2 6 10 9 5 9 2 1 4 1 5 5 5 5 8 7 4 2 8 6 10 7 3 2 8 9 6 8 5 2 9 6 10 8 6 4 9 9 4 2 9 10 7 5 4 4 4 9 7 1 5 9 9 9 10 8 8 7 5 4 1 4 1 10 3 6 5 1 6 10 5 7 10 3 3 5 8 8 6 5 9 2 3 9 10 4 7 9 1 3 2 1 6 2 2 1 9 6 1 7 5 7 3 6 9 7 3 9 5 8 3 5 1 7 3 10 10 1 9 2 4 2 2 1 4 5 1 4 10 2 10 7 10 4 4 9 1 6 7 7 5 1 1 5 7 3 7 8 6 7 10 9 8 3 9 3 10 10 7 1 3 8 7 2 4 3 2 6 10 10 2 5 10 2 1 8 6 9 8 1 5 9 1 5 3 10 7 2 1 5 3 3 3 1 6 6 3 10 1 3 9 4 9 1 5 1 10 2 10 7 3 6 5 5 10 10 4 7 1 6 1 3 10 5 4 6 2 8 5 4 2 5 7 10 5 3 3 7 5 2 3 9 9 10 3 9 9 9 7 9 4 9 4 4 4 9 1 5 8 7 9 10 1 7 9 8 10 1 4 4 4 8 4 3 7 6 3 7 6 9 8 10 7 1 5 2 1 5 9 8 1 9 7 3 5 8 10 4 10 3 9 4 1 2 8 9 10 2 6 5 10 3 6 8 5 10 10 5 6 10 4 6 8 1 9 2 10 10 8 9 3 6 4 5 10 1 3 1 2 10 7 3 2 3 1 8 4 2 2 10 1 6 7 8 8 5 1 7 5 8 5 9 6 9 3 7 1 7 7 5 7 3 9 10 7 1 8 1 2 1 2 4 8 8 3 7 5 6 3 1 3 10 1 10 10 5 6 2 1 4 8 9 9 7 1 5 7 8 7 1 10 8 6 10 8 9 6 4 4 9 4 8 10 4 8 9 8 5 2 10 1 10 9 9 6 9 5 4 8 2 4 9 1 10 8 10 10 4 3 5 4 8 2 3 3 1 3 2 8 6 2 8 5 2 8 2 2 2 8 1 5 1 9 6 2 7 7 3 2 10 7 5 9 1 9 2 1 3 3 10 8 6 7 5 7 4 8 10 8 5 10 2 8 1 7 1 9 6 4 10 5 2 6 5 2 6 6 5 10 9 4 9 6 3 3 3 8 1 4 5 7 4 7 4 4 5 5 4 10 8 3 6 9 10 1 3 5 8 7 6 8 2 4 4 4 9 6 2 1 9 8 7 4 6 1 9 1 5 2 2 4 6 10 4 5 2 6 1 9 4 6 7 6 10 10 1 8 7 4 8 7 2 6 1 7 6 1 9 2 3 3 7 10 2 1 5 3 8 5 1 4 3 9 1 4 8 1 1 4 5 10 3 8 5 3 6 3 5 5 4 9 7 1 9 10 3 3 4 2 9 4 5 3 3 5 6 2 8 6 8 2 7 10 9 2 4 4 4 8 10 9 7 8 1 5 9 5 9 2 7 9 6 3 2 10 10 7 1 7 5 10 10 1 9 10 4 2 5 9 10 7 8 8 4 8 2 3 3 2 6 1 10 1 5 1 2 4 8 5 2 2 4 1 4 3 2 8 6 7 6 5 3 3 2 8 3 5 2 7 6 8 9 6 10 3 8 4 7 5 8 10 8 3 3 2 7 4 1 3 2 6 7 6 1 1 4 5 7 7 1 3 6 9 10 6 1 7 1 9 4 8 9 3 2 1 6 9 6 6 1 8 4 9 3 4 1 8 8 8 7 10 2 2 10 1 9 10 9 9 1 4 8 1 7 10 3 2 8 8 10 8 7 3 8 2 6 9 9 5 6 7 6 7 9 5 9 9 4 9 8 6 3 5 6 1 6 8 4 3 8 3 2 4 5 10 5 2 10 6 8 5 2 3 3 2 7 3 1 3 3 10 8 5 4 6 7 2 5 1 4 2 3 8 8 10 9 2 1 8 7 1 2 1 3 6 2 10 8 2 4 2 3 1 7 9 8 5 10 3 5 5 6 10 2 3 9 10 7 1 7 5 1 10 5 6 5 7 7 4 10 10 6 3 2 4 1 10 10 2 4 5 6 9 6 8 4 4 9 10 6 6 4 9 7 1 4 4 7 2 9 8 1 6 2 2 1 4 1 1 5 6 5 3 5 10 10 10 5 10 9 2 7 4 10 4 4 5 7 2 6 7 2 8 3 3 10 3 9 2 3 5 8 9 7 4 8 8 3 4 8 3 6 4 8 7 9 2 4 7 5 1 4 6 9 6 1 10 10 9 1 3 5 10 1 2 3 1 1 7 4 10 1 9 4 9 8 4 2 1 1 6 1 4 2 9 1 2 8 10 2 1 4 6 10 5 9 5 7 10 3 10 1 4 1 4 4 8 10 5 10 10 2 10 5 3 1 7 6 10 6 7 2 2 5 4 8 3 8 4 4 10 3 5 5 5 10 10 4 9 4 3 10 6 5 4 10 5 10 6 6 8 4 10 1 8 3 8 3 10 1 6 1 5 2 8 10 4 7 3 2 3 6 4 10 2 9 9 8 1 6 3 8 10 2 10 7 4 7 9 5 7 7 8 3 8 5 2 1 1 7 5 5 4 8 4 5 8 3 2 8 10 6 7 9 10 6 6 3 2 6 10 10 2 7 3 2 1 6 4 3 2 10 8 7 9 3 1 7 5 4 4 7 10 1 5 9 6 2 3 10 8 4 9 1 10 3 4 2 9 8 5 2 7 4 9 6 6 1 2 3 7 5 9 6 7 3 6 3 7 10 2 4 4 2 6 3 7 10 5 5 7 1 8 5 4 8 10 1 1 3 3 7 10 1 4 6 6 9 8 2 8 1 7 1 5 2 4 1 3 10 7 1 10 4 6 5 4 7 5 4 2 10 2 1 10 5 6 7 3 6 10 2 8 8 5 2 2 10 2 4 9 8 5 10 4 2 4 9 8 10 4 9 9 5 1 1 9 7 9 3 4 9 6 1 6 10 3 7 9 6 3 9

**Output:**

4-157

7-144

8-148

6-134

3-148

10-172

2-149

1-159

9-142

5-147

**Testcase 8**

**Input:**

10

884 887 778 416 294 836 887 493 1150 422

**Output:**

884-1

887-2

778-1

416-1

294-1

836-1

493-1

1150-1

422-1

**Testcase 9**

**Input:**

10

4 7 18 16 14 16 7 13 10 2

**Output:**

4-1

7-2

18-1

16-2

14-1

13-1

10-1

2-1

**Program 15**

**Sub Topic:**Algorithm

***Odd numbers in a range***

*﻿*To find the odd numbers in between the range.

**Input Format**

Input represents two integers start and end range

**Output Format**

Print the odd numbers separated by space

**Constraints**

1<= start<=end<=1000000

**Sample Input 1**

2 15

**Sample Output 1**

3 5 7 9 11 13

**Solution:**

#include<stdio.h>

int main()

{

int start , end,ctr;

scanf("%d %d",&start,&end);

if( start % 2 == 0 )

start++;

for( ctr = start ; ctr < end ; ctr+=2 )

printf("%d ", ctr);

return 0;

}

**Testcase 1:**

**Input:**

2 15

**Output:**

3 5 7 9 11 13

**Testcase 2:**

**Input:**

10 50

**Output:**

11 13 15 17 19 21 23 25 27 29 31 33 35 37 39 41 43 45 47 49

**Testcase 3:**

**Input:**

100 525

**Output:**

101 103 105 107 109 111 113 115 117 119 121 123 125 127 129 131 133 135 137 139 141 143 145 147 149 151 153 155 157 159 161 163 165 167 169 171 173 175 177 179 181 183 185 187 189 191 193 195 197 199 201 203 205 207 209 211 213 215 217 219 221 223 225 227 229 231 233 235 237 239 241 243 245 247 249 251 253 255 257 259 261 263 265 267 269 271 273 275 277 279 281 283 285 287 289 291 293 295 297 299 301 303 305 307 309 311 313 315 317 319 321 323 325 327 329 331 333 335 337 339 341 343 345 347 349 351 353 355 357 359 361 363 365 367 369 371 373 375 377 379 381 383 385 387 389 391 393 395 397 399 401 403 405 407 409 411 413 415 417 419 421 423 425 427 429 431 433 435 437 439 441 443 445 447 449 451 453 455 457 459 461 463 465 467 469 471 473 475 477 479 481 483 485 487 489 491 493 495 497 499 501 503 505 507 509 511 513 515 517 519 521 523

**Testcase 4:**

**Input:**

1000 1256

**Output:**

1001 1003 1005 1007 1009 1011 1013 1015 1017 1019 1021 1023 1025 1027 1029 1031 1033 1035 1037 1039 1041 1043 1045 1047 1049 1051 1053 1055 1057 1059 1061 1063 1065 1067 1069 1071 1073 1075 1077 1079 1081 1083 1085 1087 1089 1091 1093 1095 1097 1099 1101 1103 1105 1107 1109 1111 1113 1115 1117 1119 1121 1123 1125 1127 1129 1131 1133 1135 1137 1139 1141 1143 1145 1147 1149 1151 1153 1155 1157 1159 1161 1163 1165 1167 1169 1171 1173 1175 1177 1179 1181 1183 1185 1187 1189 1191 1193 1195 1197 1199 1201 1203 1205 1207 1209 1211 1213 1215 1217 1219 1221 1223 1225 1227 1229 1231 1233 1235 1237 1239 1241 1243 1245 1247 1249 1251 1253 1255

**Testcase 5:**

**Input:**

555 1000

**Output:**

555 557 559 561 563 565 567 569 571 573 575 577 579 581 583 585 587 589 591 593 595 597 599 601 603 605 607 609 611 613 615 617 619 621 623 625 627 629 631 633 635 637 639 641 643 645 647 649 651 653 655 657 659 661 663 665 667 669 671 673 675 677 679 681 683 685 687 689 691 693 695 697 699 701 703 705 707 709 711 713 715 717 719 721 723 725 727 729 731 733 735 737 739 741 743 745 747 749 751 753 755 757 759 761 763 765 767 769 771 773 775 777 779 781 783 785 787 789 791 793 795 797 799 801 803 805 807 809 811 813 815 817 819 821 823 825 827 829 831 833 835 837 839 841 843 845 847 849 851 853 855 857 859 861 863 865 867 869 871 873 875 877 879 881 883 885 887 889 891 893 895 897 899 901 903 905 907 909 911 913 915 917 919 921 923 925 927 929 931 933 935 937 939 941 943 945 947 949 951 953 955 957 959 961 963 965 967 969 971 973 975 977 979 981 983 985 987 989 991 993 995 997 999

**Testcase 6:**

**Input:**

100 1000

**Output:**

101 103 105 107 109 111 113 115 117 119 121 123 125 127 129 131 133 135 137 139 141 143 145 147 149 151 153 155 157 159 161 163 165 167 169 171 173 175 177 179 181 183 185 187 189 191 193 195 197 199 201 203 205 207 209 211 213 215 217 219 221 223 225 227 229 231 233 235 237 239 241 243 245 247 249 251 253 255 257 259 261 263 265 267 269 271 273 275 277 279 281 283 285 287 289 291 293 295 297 299 301 303 305 307 309 311 313 315 317 319 321 323 325 327 329 331 333 335 337 339 341 343 345 347 349 351 353 355 357 359 361 363 365 367 369 371 373 375 377 379 381 383 385 387 389 391 393 395 397 399 401 403 405 407 409 411 413 415 417 419 421 423 425 427 429 431 433 435 437 439 441 443 445 447 449 451 453 455 457 459 461 463 465 467 469 471 473 475 477 479 481 483 485 487 489 491 493 495 497 499 501 503 505 507 509 511 513 515 517 519 521 523 525 527 529 531 533 535 537 539 541 543 545 547 549 551 553 555 557 559 561 563 565 567 569 571 573 575 577 579 581 583 585 587 589 591 593 595 597 599 601 603 605 607 609 611 613 615 617 619 621 623 625 627 629 631 633 635 637 639 641 643 645 647 649 651 653 655 657 659 661 663 665 667 669 671 673 675 677 679 681 683 685 687 689 691 693 695 697 699 701 703 705 707 709 711 713 715 717 719 721 723 725 727 729 731 733 735 737 739 741 743 745 747 749 751 753 755 757 759 761 763 765 767 769 771 773 775 777 779 781 783 785 787 789 791 793 795 797 799 801 803 805 807 809 811 813 815 817 819 821 823 825 827 829 831 833 835 837 839 841 843 845 847 849 851 853 855 857 859 861 863 865 867 869 871 873 875 877 879 881 883 885 887 889 891 893 895 897 899 901 903 905 907 909 911 913 915 917 919 921 923 925 927 929 931 933 935 937 939 941 943 945 947 949 951 953 955 957 959 961 963 965 967 969 971 973 975 977 979 981 983 985 987 989 991 993 995 997 999

**Testcase 7:**

**Input:**

1500 5000

**Output:**

1501 1503 1505 1507 1509 1511 1513 1515 1517 1519 1521 1523 1525 1527 1529 1531 1533 1535 1537 1539 1541 1543 1545 1547 1549 1551 1553 1555 1557 1559 1561 1563 1565 1567 1569 1571 1573 1575 1577 1579 1581 1583 1585 1587 1589 1591 1593 1595 1597 1599 1601 1603 1605 1607 1609 1611 1613 1615 1617 1619 1621 1623 1625 1627 1629 1631 1633 1635 1637 1639 1641 1643 1645 1647 1649 1651 1653 1655 1657 1659 1661 1663 1665 1667 1669 1671 1673 1675 1677 1679 1681 1683 1685 1687 1689 1691 1693 1695 1697 1699 1701 1703 1705 1707 1709 1711 1713 1715 1717 1719 1721 1723 1725 1727 1729 1731 1733 1735 1737 1739 1741 1743 1745 1747 1749 1751 1753 1755 1757 1759 1761 1763 1765 1767 1769 1771 1773 1775 1777 1779 1781 1783 1785 1787 1789 1791 1793 1795 1797 1799 1801 1803 1805 1807 1809 1811 1813 1815 1817 1819 1821 1823 1825 1827 1829 1831 1833 1835 1837 1839 1841 1843 1845 1847 1849 1851 1853 1855 1857 1859 1861 1863 1865 1867 1869 1871 1873 1875 1877 1879 1881 1883 1885 1887 1889 1891 1893 1895 1897 1899 1901 1903 1905 1907 1909 1911 1913 1915 1917 1919 1921 1923 1925 1927 1929 1931 1933 1935 1937 1939 1941 1943 1945 1947 1949 1951 1953 1955 1957 1959 1961 1963 1965 1967 1969 1971 1973 1975 1977 1979 1981 1983 1985 1987 1989 1991 1993 1995 1997 1999 2001 2003 2005 2007 2009 2011 2013 2015 2017 2019 2021 2023 2025 2027 2029 2031 2033 2035 2037 2039 2041 2043 2045 2047 2049 2051 2053 2055 2057 2059 2061 2063 2065 2067 2069 2071 2073 2075 2077 2079 2081 2083 2085 2087 2089 2091 2093 2095 2097 2099 2101 2103 2105 2107 2109 2111 2113 2115 2117 2119 2121 2123 2125 2127 2129 2131 2133 2135 2137 2139 2141 2143 2145 2147 2149 2151 2153 2155 2157 2159 2161 2163 2165 2167 2169 2171 2173 2175 2177 2179 2181 2183 2185 2187 2189 2191 2193 2195 2197 2199 2201 2203 2205 2207 2209 2211 2213 2215 2217 2219 2221 2223 2225 2227 2229 2231 2233 2235 2237 2239 2241 2243 2245 2247 2249 2251 2253 2255 2257 2259 2261 2263 2265 2267 2269 2271 2273 2275 2277 2279 2281 2283 2285 2287 2289 2291 2293 2295 2297 2299 2301 2303 2305 2307 2309 2311 2313 2315 2317 2319 2321 2323 2325 2327 2329 2331 2333 2335 2337 2339 2341 2343 2345 2347 2349 2351 2353 2355 2357 2359 2361 2363 2365 2367 2369 2371 2373 2375 2377 2379 2381 2383 2385 2387 2389 2391 2393 2395 2397 2399 2401 2403 2405 2407 2409 2411 2413 2415 2417 2419 2421 2423 2425 2427 2429 2431 2433 2435 2437 2439 2441 2443 2445 2447 2449 2451 2453 2455 2457 2459 2461 2463 2465 2467 2469 2471 2473 2475 2477 2479 2481 2483 2485 2487 2489 2491 2493 2495 2497 2499 2501 2503 2505 2507 2509 2511 2513 2515 2517 2519 2521 2523 2525 2527 2529 2531 2533 2535 2537 2539 2541 2543 2545 2547 2549 2551 2553 2555 2557 2559 2561 2563 2565 2567 2569 2571 2573 2575 2577 2579 2581 2583 2585 2587 2589 2591 2593 2595 2597 2599 2601 2603 2605 2607 2609 2611 2613 2615 2617 2619 2621 2623 2625 2627 2629 2631 2633 2635 2637 2639 2641 2643 2645 2647 2649 2651 2653 2655 2657 2659 2661 2663 2665 2667 2669 2671 2673 2675 2677 2679 2681 2683 2685 2687 2689 2691 2693 2695 2697 2699 2701 2703 2705 2707 2709 2711 2713 2715 2717 2719 2721 2723 2725 2727 2729 2731 2733 2735 2737 2739 2741 2743 2745 2747 2749 2751 2753 2755 2757 2759 2761 2763 2765 2767 2769 2771 2773 2775 2777 2779 2781 2783 2785 2787 2789 2791 2793 2795 2797 2799 2801 2803 2805 2807 2809 2811 2813 2815 2817 2819 2821 2823 2825 2827 2829 2831 2833 2835 2837 2839 2841 2843 2845 2847 2849 2851 2853 2855 2857 2859 2861 2863 2865 2867 2869 2871 2873 2875 2877 2879 2881 2883 2885 2887 2889 2891 2893 2895 2897 2899 2901 2903 2905 2907 2909 2911 2913 2915 2917 2919 2921 2923 2925 2927 2929 2931 2933 2935 2937 2939 2941 2943 2945 2947 2949 2951 2953 2955 2957 2959 2961 2963 2965 2967 2969 2971 2973 2975 2977 2979 2981 2983 2985 2987 2989 2991 2993 2995 2997 2999 3001 3003 3005 3007 3009 3011 3013 3015 3017 3019 3021 3023 3025 3027 3029 3031 3033 3035 3037 3039 3041 3043 3045 3047 3049 3051 3053 3055 3057 3059 3061 3063 3065 3067 3069 3071 3073 3075 3077 3079 3081 3083 3085 3087 3089 3091 3093 3095 3097 3099 3101 3103 3105 3107 3109 3111 3113 3115 3117 3119 3121 3123 3125 3127 3129 3131 3133 3135 3137 3139 3141 3143 3145 3147 3149 3151 3153 3155 3157 3159 3161 3163 3165 3167 3169 3171 3173 3175 3177 3179 3181 3183 3185 3187 3189 3191 3193 3195 3197 3199 3201 3203 3205 3207 3209 3211 3213 3215 3217 3219 3221 3223 3225 3227 3229 3231 3233 3235 3237 3239 3241 3243 3245 3247 3249 3251 3253 3255 3257 3259 3261 3263 3265 3267 3269 3271 3273 3275 3277 3279 3281 3283 3285 3287 3289 3291 3293 3295 3297 3299 3301 3303 3305 3307 3309 3311 3313 3315 3317 3319 3321 3323 3325 3327 3329 3331 3333 3335 3337 3339 3341 3343 3345 3347 3349 3351 3353 3355 3357 3359 3361 3363 3365 3367 3369 3371 3373 3375 3377 3379 3381 3383 3385 3387 3389 3391 3393 3395 3397 3399 3401 3403 3405 3407 3409 3411 3413 3415 3417 3419 3421 3423 3425 3427 3429 3431 3433 3435 3437 3439 3441 3443 3445 3447 3449 3451 3453 3455 3457 3459 3461 3463 3465 3467 3469 3471 3473 3475 3477 3479 3481 3483 3485 3487 3489 3491 3493 3495 3497 3499 3501 3503 3505 3507 3509 3511 3513 3515 3517 3519 3521 3523 3525 3527 3529 3531 3533 3535 3537 3539 3541 3543 3545 3547 3549 3551 3553 3555 3557 3559 3561 3563 3565 3567 3569 3571 3573 3575 3577 3579 3581 3583 3585 3587 3589 3591 3593 3595 3597 3599 3601 3603 3605 3607 3609 3611 3613 3615 3617 3619 3621 3623 3625 3627 3629 3631 3633 3635 3637 3639 3641 3643 3645 3647 3649 3651 3653 3655 3657 3659 3661 3663 3665 3667 3669 3671 3673 3675 3677 3679 3681 3683 3685 3687 3689 3691 3693 3695 3697 3699 3701 3703 3705 3707 3709 3711 3713 3715 3717 3719 3721 3723 3725 3727 3729 3731 3733 3735 3737 3739 3741 3743 3745 3747 3749 3751 3753 3755 3757 3759 3761 3763 3765 3767 3769 3771 3773 3775 3777 3779 3781 3783 3785 3787 3789 3791 3793 3795 3797 3799 3801 3803 3805 3807 3809 3811 3813 3815 3817 3819 3821 3823 3825 3827 3829 3831 3833 3835 3837 3839 3841 3843 3845 3847 3849 3851 3853 3855 3857 3859 3861 3863 3865 3867 3869 3871 3873 3875 3877 3879 3881 3883 3885 3887 3889 3891 3893 3895 3897 3899 3901 3903 3905 3907 3909 3911 3913 3915 3917 3919 3921 3923 3925 3927 3929 3931 3933 3935 3937 3939 3941 3943 3945 3947 3949 3951 3953 3955 3957 3959 3961 3963 3965 3967 3969 3971 3973 3975 3977 3979 3981 3983 3985 3987 3989 3991 3993 3995 3997 3999 4001 4003 4005 4007 4009 4011 4013 4015 4017 4019 4021 4023 4025 4027 4029 4031 4033 4035 4037 4039 4041 4043 4045 4047 4049 4051 4053 4055 4057 4059 4061 4063 4065 4067 4069 4071 4073 4075 4077 4079 4081 4083 4085 4087 4089 4091 4093 4095 4097 4099 4101 4103 4105 4107 4109 4111 4113 4115 4117 4119 4121 4123 4125 4127 4129 4131 4133 4135 4137 4139 4141 4143 4145 4147 4149 4151 4153 4155 4157 4159 4161 4163 4165 4167 4169 4171 4173 4175 4177 4179 4181 4183 4185 4187 4189 4191 4193 4195 4197 4199 4201 4203 4205 4207 4209 4211 4213 4215 4217 4219 4221 4223 4225 4227 4229 4231 4233 4235 4237 4239 4241 4243 4245 4247 4249 4251 4253 4255 4257 4259 4261 4263 4265 4267 4269 4271 4273 4275 4277 4279 4281 4283 4285 4287 4289 4291 4293 4295 4297 4299 4301 4303 4305 4307 4309 4311 4313 4315 4317 4319 4321 4323 4325 4327 4329 4331 4333 4335 4337 4339 4341 4343 4345 4347 4349 4351 4353 4355 4357 4359 4361 4363 4365 4367 4369 4371 4373 4375 4377 4379 4381 4383 4385 4387 4389 4391 4393 4395 4397 4399 4401 4403 4405 4407 4409 4411 4413 4415 4417 4419 4421 4423 4425 4427 4429 4431 4433 4435 4437 4439 4441 4443 4445 4447 4449 4451 4453 4455 4457 4459 4461 4463 4465 4467 4469 4471 4473 4475 4477 4479 4481 4483 4485 4487 4489 4491 4493 4495 4497 4499 4501 4503 4505 4507 4509 4511 4513 4515 4517 4519 4521 4523 4525 4527 4529 4531 4533 4535 4537 4539 4541 4543 4545 4547 4549 4551 4553 4555 4557 4559 4561 4563 4565 4567 4569 4571 4573 4575 4577 4579 4581 4583 4585 4587 4589 4591 4593 4595 4597 4599 4601 4603 4605 4607 4609 4611 4613 4615 4617 4619 4621 4623 4625 4627 4629 4631 4633 4635 4637 4639 4641 4643 4645 4647 4649 4651 4653 4655 4657 4659 4661 4663 4665 4667 4669 4671 4673 4675 4677 4679 4681 4683 4685 4687 4689 4691 4693 4695 4697 4699 4701 4703 4705 4707 4709 4711 4713 4715 4717 4719 4721 4723 4725 4727 4729 4731 4733 4735 4737 4739 4741 4743 4745 4747 4749 4751 4753 4755 4757 4759 4761 4763 4765 4767 4769 4771 4773 4775 4777 4779 4781 4783 4785 4787 4789 4791 4793 4795 4797 4799 4801 4803 4805 4807 4809 4811 4813 4815 4817 4819 4821 4823 4825 4827 4829 4831 4833 4835 4837 4839 4841 4843 4845 4847 4849 4851 4853 4855 4857 4859 4861 4863 4865 4867 4869 4871 4873 4875 4877 4879 4881 4883 4885 4887 4889 4891 4893 4895 4897 4899 4901 4903 4905 4907 4909 4911 4913 4915 4917 4919 4921 4923 4925 4927 4929 4931 4933 4935 4937 4939 4941 4943 4945 4947 4949 4951 4953 4955 4957 4959 4961 4963 4965 4967 4969 4971 4973 4975 4977 4979 4981 4983 4985 4987 4989 4991 4993 4995 4997 4999

**Testcase 8:**

**Input:**

5000 8000

**Output:**

5001 5003 5005 5007 5009 5011 5013 5015 5017 5019 5021 5023 5025 5027 5029 5031 5033 5035 5037 5039 5041 5043 5045 5047 5049 5051 5053 5055 5057 5059 5061 5063 5065 5067 5069 5071 5073 5075 5077 5079 5081 5083 5085 5087 5089 5091 5093 5095 5097 5099 5101 5103 5105 5107 5109 5111 5113 5115 5117 5119 5121 5123 5125 5127 5129 5131 5133 5135 5137 5139 5141 5143 5145 5147 5149 5151 5153 5155 5157 5159 5161 5163 5165 5167 5169 5171 5173 5175 5177 5179 5181 5183 5185 5187 5189 5191 5193 5195 5197 5199 5201 5203 5205 5207 5209 5211 5213 5215 5217 5219 5221 5223 5225 5227 5229 5231 5233 5235 5237 5239 5241 5243 5245 5247 5249 5251 5253 5255 5257 5259 5261 5263 5265 5267 5269 5271 5273 5275 5277 5279 5281 5283 5285 5287 5289 5291 5293 5295 5297 5299 5301 5303 5305 5307 5309 5311 5313 5315 5317 5319 5321 5323 5325 5327 5329 5331 5333 5335 5337 5339 5341 5343 5345 5347 5349 5351 5353 5355 5357 5359 5361 5363 5365 5367 5369 5371 5373 5375 5377 5379 5381 5383 5385 5387 5389 5391 5393 5395 5397 5399 5401 5403 5405 5407 5409 5411 5413 5415 5417 5419 5421 5423 5425 5427 5429 5431 5433 5435 5437 5439 5441 5443 5445 5447 5449 5451 5453 5455 5457 5459 5461 5463 5465 5467 5469 5471 5473 5475 5477 5479 5481 5483 5485 5487 5489 5491 5493 5495 5497 5499 5501 5503 5505 5507 5509 5511 5513 5515 5517 5519 5521 5523 5525 5527 5529 5531 5533 5535 5537 5539 5541 5543 5545 5547 5549 5551 5553 5555 5557 5559 5561 5563 5565 5567 5569 5571 5573 5575 5577 5579 5581 5583 5585 5587 5589 5591 5593 5595 5597 5599 5601 5603 5605 5607 5609 5611 5613 5615 5617 5619 5621 5623 5625 5627 5629 5631 5633 5635 5637 5639 5641 5643 5645 5647 5649 5651 5653 5655 5657 5659 5661 5663 5665 5667 5669 5671 5673 5675 5677 5679 5681 5683 5685 5687 5689 5691 5693 5695 5697 5699 5701 5703 5705 5707 5709 5711 5713 5715 5717 5719 5721 5723 5725 5727 5729 5731 5733 5735 5737 5739 5741 5743 5745 5747 5749 5751 5753 5755 5757 5759 5761 5763 5765 5767 5769 5771 5773 5775 5777 5779 5781 5783 5785 5787 5789 5791 5793 5795 5797 5799 5801 5803 5805 5807 5809 5811 5813 5815 5817 5819 5821 5823 5825 5827 5829 5831 5833 5835 5837 5839 5841 5843 5845 5847 5849 5851 5853 5855 5857 5859 5861 5863 5865 5867 5869 5871 5873 5875 5877 5879 5881 5883 5885 5887 5889 5891 5893 5895 5897 5899 5901 5903 5905 5907 5909 5911 5913 5915 5917 5919 5921 5923 5925 5927 5929 5931 5933 5935 5937 5939 5941 5943 5945 5947 5949 5951 5953 5955 5957 5959 5961 5963 5965 5967 5969 5971 5973 5975 5977 5979 5981 5983 5985 5987 5989 5991 5993 5995 5997 5999 6001 6003 6005 6007 6009 6011 6013 6015 6017 6019 6021 6023 6025 6027 6029 6031 6033 6035 6037 6039 6041 6043 6045 6047 6049 6051 6053 6055 6057 6059 6061 6063 6065 6067 6069 6071 6073 6075 6077 6079 6081 6083 6085 6087 6089 6091 6093 6095 6097 6099 6101 6103 6105 6107 6109 6111 6113 6115 6117 6119 6121 6123 6125 6127 6129 6131 6133 6135 6137 6139 6141 6143 6145 6147 6149 6151 6153 6155 6157 6159 6161 6163 6165 6167 6169 6171 6173 6175 6177 6179 6181 6183 6185 6187 6189 6191 6193 6195 6197 6199 6201 6203 6205 6207 6209 6211 6213 6215 6217 6219 6221 6223 6225 6227 6229 6231 6233 6235 6237 6239 6241 6243 6245 6247 6249 6251 6253 6255 6257 6259 6261 6263 6265 6267 6269 6271 6273 6275 6277 6279 6281 6283 6285 6287 6289 6291 6293 6295 6297 6299 6301 6303 6305 6307 6309 6311 6313 6315 6317 6319 6321 6323 6325 6327 6329 6331 6333 6335 6337 6339 6341 6343 6345 6347 6349 6351 6353 6355 6357 6359 6361 6363 6365 6367 6369 6371 6373 6375 6377 6379 6381 6383 6385 6387 6389 6391 6393 6395 6397 6399 6401 6403 6405 6407 6409 6411 6413 6415 6417 6419 6421 6423 6425 6427 6429 6431 6433 6435 6437 6439 6441 6443 6445 6447 6449 6451 6453 6455 6457 6459 6461 6463 6465 6467 6469 6471 6473 6475 6477 6479 6481 6483 6485 6487 6489 6491 6493 6495 6497 6499 6501 6503 6505 6507 6509 6511 6513 6515 6517 6519 6521 6523 6525 6527 6529 6531 6533 6535 6537 6539 6541 6543 6545 6547 6549 6551 6553 6555 6557 6559 6561 6563 6565 6567 6569 6571 6573 6575 6577 6579 6581 6583 6585 6587 6589 6591 6593 6595 6597 6599 6601 6603 6605 6607 6609 6611 6613 6615 6617 6619 6621 6623 6625 6627 6629 6631 6633 6635 6637 6639 6641 6643 6645 6647 6649 6651 6653 6655 6657 6659 6661 6663 6665 6667 6669 6671 6673 6675 6677 6679 6681 6683 6685 6687 6689 6691 6693 6695 6697 6699 6701 6703 6705 6707 6709 6711 6713 6715 6717 6719 6721 6723 6725 6727 6729 6731 6733 6735 6737 6739 6741 6743 6745 6747 6749 6751 6753 6755 6757 6759 6761 6763 6765 6767 6769 6771 6773 6775 6777 6779 6781 6783 6785 6787 6789 6791 6793 6795 6797 6799 6801 6803 6805 6807 6809 6811 6813 6815 6817 6819 6821 6823 6825 6827 6829 6831 6833 6835 6837 6839 6841 6843 6845 6847 6849 6851 6853 6855 6857 6859 6861 6863 6865 6867 6869 6871 6873 6875 6877 6879 6881 6883 6885 6887 6889 6891 6893 6895 6897 6899 6901 6903 6905 6907 6909 6911 6913 6915 6917 6919 6921 6923 6925 6927 6929 6931 6933 6935 6937 6939 6941 6943 6945 6947 6949 6951 6953 6955 6957 6959 6961 6963 6965 6967 6969 6971 6973 6975 6977 6979 6981 6983 6985 6987 6989 6991 6993 6995 6997 6999 7001 7003 7005 7007 7009 7011 7013 7015 7017 7019 7021 7023 7025 7027 7029 7031 7033 7035 7037 7039 7041 7043 7045 7047 7049 7051 7053 7055 7057 7059 7061 7063 7065 7067 7069 7071 7073 7075 7077 7079 7081 7083 7085 7087 7089 7091 7093 7095 7097 7099 7101 7103 7105 7107 7109 7111 7113 7115 7117 7119 7121 7123 7125 7127 7129 7131 7133 7135 7137 7139 7141 7143 7145 7147 7149 7151 7153 7155 7157 7159 7161 7163 7165 7167 7169 7171 7173 7175 7177 7179 7181 7183 7185 7187 7189 7191 7193 7195 7197 7199 7201 7203 7205 7207 7209 7211 7213 7215 7217 7219 7221 7223 7225 7227 7229 7231 7233 7235 7237 7239 7241 7243 7245 7247 7249 7251 7253 7255 7257 7259 7261 7263 7265 7267 7269 7271 7273 7275 7277 7279 7281 7283 7285 7287 7289 7291 7293 7295 7297 7299 7301 7303 7305 7307 7309 7311 7313 7315 7317 7319 7321 7323 7325 7327 7329 7331 7333 7335 7337 7339 7341 7343 7345 7347 7349 7351 7353 7355 7357 7359 7361 7363 7365 7367 7369 7371 7373 7375 7377 7379 7381 7383 7385 7387 7389 7391 7393 7395 7397 7399 7401 7403 7405 7407 7409 7411 7413 7415 7417 7419 7421 7423 7425 7427 7429 7431 7433 7435 7437 7439 7441 7443 7445 7447 7449 7451 7453 7455 7457 7459 7461 7463 7465 7467 7469 7471 7473 7475 7477 7479 7481 7483 7485 7487 7489 7491 7493 7495 7497 7499 7501 7503 7505 7507 7509 7511 7513 7515 7517 7519 7521 7523 7525 7527 7529 7531 7533 7535 7537 7539 7541 7543 7545 7547 7549 7551 7553 7555 7557 7559 7561 7563 7565 7567 7569 7571 7573 7575 7577 7579 7581 7583 7585 7587 7589 7591 7593 7595 7597 7599 7601 7603 7605 7607 7609 7611 7613 7615 7617 7619 7621 7623 7625 7627 7629 7631 7633 7635 7637 7639 7641 7643 7645 7647 7649 7651 7653 7655 7657 7659 7661 7663 7665 7667 7669 7671 7673 7675 7677 7679 7681 7683 7685 7687 7689 7691 7693 7695 7697 7699 7701 7703 7705 7707 7709 7711 7713 7715 7717 7719 7721 7723 7725 7727 7729 7731 7733 7735 7737 7739 7741 7743 7745 7747 7749 7751 7753 7755 7757 7759 7761 7763 7765 7767 7769 7771 7773 7775 7777 7779 7781 7783 7785 7787 7789 7791 7793 7795 7797 7799 7801 7803 7805 7807 7809 7811 7813 7815 7817 7819 7821 7823 7825 7827 7829 7831 7833 7835 7837 7839 7841 7843 7845 7847 7849 7851 7853 7855 7857 7859 7861 7863 7865 7867 7869 7871 7873 7875 7877 7879 7881 7883 7885 7887 7889 7891 7893 7895 7897 7899 7901 7903 7905 7907 7909 7911 7913 7915 7917 7919 7921 7923 7925 7927 7929 7931 7933 7935 7937 7939 7941 7943 7945 7947 7949 7951 7953 7955 7957 7959 7961 7963 7965 7967 7969 7971 7973 7975 7977 7979 7981 7983 7985 7987 7989 7991 7993 7995 7997 7999

**Testcase 9**

**Input:**

8235 9315

**Output:**

8235 8237 8239 8241 8243 8245 8247 8249 8251 8253 8255 8257 8259 8261 8263 8265 8267 8269 8271 8273 8275 8277 8279 8281 8283 8285 8287 8289 8291 8293 8295 8297 8299 8301 8303 8305 8307 8309 8311 8313 8315 8317 8319 8321 8323 8325 8327 8329 8331 8333 8335 8337 8339 8341 8343 8345 8347 8349 8351 8353 8355 8357 8359 8361 8363 8365 8367 8369 8371 8373 8375 8377 8379 8381 8383 8385 8387 8389 8391 8393 8395 8397 8399 8401 8403 8405 8407 8409 8411 8413 8415 8417 8419 8421 8423 8425 8427 8429 8431 8433 8435 8437 8439 8441 8443 8445 8447 8449 8451 8453 8455 8457 8459 8461 8463 8465 8467 8469 8471 8473 8475 8477 8479 8481 8483 8485 8487 8489 8491 8493 8495 8497 8499 8501 8503 8505 8507 8509 8511 8513 8515 8517 8519 8521 8523 8525 8527 8529 8531 8533 8535 8537 8539 8541 8543 8545 8547 8549 8551 8553 8555 8557 8559 8561 8563 8565 8567 8569 8571 8573 8575 8577 8579 8581 8583 8585 8587 8589 8591 8593 8595 8597 8599 8601 8603 8605 8607 8609 8611 8613 8615 8617 8619 8621 8623 8625 8627 8629 8631 8633 8635 8637 8639 8641 8643 8645 8647 8649 8651 8653 8655 8657 8659 8661 8663 8665 8667 8669 8671 8673 8675 8677 8679 8681 8683 8685 8687 8689 8691 8693 8695 8697 8699 8701 8703 8705 8707 8709 8711 8713 8715 8717 8719 8721 8723 8725 8727 8729 8731 8733 8735 8737 8739 8741 8743 8745 8747 8749 8751 8753 8755 8757 8759 8761 8763 8765 8767 8769 8771 8773 8775 8777 8779 8781 8783 8785 8787 8789 8791 8793 8795 8797 8799 8801 8803 8805 8807 8809 8811 8813 8815 8817 8819 8821 8823 8825 8827 8829 8831 8833 8835 8837 8839 8841 8843 8845 8847 8849 8851 8853 8855 8857 8859 8861 8863 8865 8867 8869 8871 8873 8875 8877 8879 8881 8883 8885 8887 8889 8891 8893 8895 8897 8899 8901 8903 8905 8907 8909 8911 8913 8915 8917 8919 8921 8923 8925 8927 8929 8931 8933 8935 8937 8939 8941 8943 8945 8947 8949 8951 8953 8955 8957 8959 8961 8963 8965 8967 8969 8971 8973 8975 8977 8979 8981 8983 8985 8987 8989 8991 8993 8995 8997 8999 9001 9003 9005 9007 9009 9011 9013 9015 9017 9019 9021 9023 9025 9027 9029 9031 9033 9035 9037 9039 9041 9043 9045 9047 9049 9051 9053 9055 9057 9059 9061 9063 9065 9067 9069 9071 9073 9075 9077 9079 9081 9083 9085 9087 9089 9091 9093 9095 9097 9099 9101 9103 9105 9107 9109 9111 9113 9115 9117 9119 9121 9123 9125 9127 9129 9131 9133 9135 9137 9139 9141 9143 9145 9147 9149 9151 9153 9155 9157 9159 9161 9163 9165 9167 9169 9171 9173 9175 9177 9179 9181 9183 9185 9187 9189 9191 9193 9195 9197 9199 9201 9203 9205 9207 9209 9211 9213 9215 9217 9219 9221 9223 9225 9227 9229 9231 9233 9235 9237 9239 9241 9243 9245 9247 9249 9251 9253 9255 9257 9259 9261 9263 9265 9267 9269 9271 9273 9275 9277 9279 9281 9283 9285 9287 9289 9291 9293 9295 9297 9299 9301 9303 9305 9307 9309 9311 9313

**Testcase 10:**

**Input:**

5 6

**Output:**

5

**Testcase 11:**

**Input:**

225 235

**Output:**

225 227 229 231 233

**Testcase 12:**

**Input:**

89 125

**Output:**

89 91 93 95 97 99 101 103 105 107 109 111 113 115 117 119 121 123

**Program 16:**

**Sub topic:** Arrays

***Pattern***

**Sample Input 1**

4

**Sample Output 1**

4444

4334

4334

4444

**Solution:**

#include<stdio.h>

int setNum(int N , int \*pow)

{

int ind , num;

num = 0 ;

\*pow = 1;

for(ind = 1 ; ind <= N/2 ; num = num \* 10 + N , (\*pow) = (\*pow) \* 10 + 1 , ind++);

(\*pow) /= 100;

return num;

}

int reverse(int num)

{

int revnum = 0;

while(num)

{

revnum = revnum \* 10 + (num % 10);

num /= 10;

}

return revnum;

}

int main()

{

int N , num , pow , row , col;

scanf("%d" , &N);

num = setNum(N , &pow);

for(row = 1 ; row <= N/2 ; row++,printf("\n"))

{

printf("%d%d",num,reverse(num));

num = num - pow;

pow /= 10;

}

for(row = 1 ; row <= N/2 ; row++,printf("\n"))

{

printf("%d%d",num,reverse(num));

pow = pow \* 10 + 1;

num = num + pow;

}

return 0;

}

**Testcase 1:**

**Input:**

4

**Output:**

4444

4334

4334

4444

**Testcase 2:**

**Input:**

6

**Output:**

666666

655556

654456

654456

655556

666666

**Testcase 3:**

**Input:**

2

**Output:**

22

22

**Testcase 4:**

**Input:**

5

**Output:**

5555

5445

5445

5555

**Testcase 5:**

**Input:**

7

**Output:**

777777

766667

765567

765567

766667

777777

**Testcase 6:**

**Input:**

8

**Output:**

88888888

87777778

87666678

87655678

87655678

87666678

87777778

88888888

**Testcase 7:**

**Input:**

9

**Output:**

99999999

98888889

98777789

98766789

98766789

98777789

98888889

99999999

**Testcase 8:**

**Input:**

3

**Output:**

33

33

**Program 17:**

**Sub Topic :** Strings

Print the string

note : no extra space

works only on odd length string

Sample Input 1

welcome

Sample Output 1

w e

e m

l o

c

l o

e m

w e

**Solution:**

#include<stdio.h>

int main()

{

char str[100];

scanf("%s" , str);

int len , space1 , space2 , row , ctr,start,end;

for(len = 0 ; str[len] ; len++);

space2 = len - 2;

space1 = 0;

start = 0;

end = len - 1;

for(row = 1 ; row <= (len/2+1) && start < end; row++, printf("\n"), space1++ , space2-=2)

{

for(ctr = 1 ; ctr <= space1 ; printf(" "), ctr++);

printf("%c",str[start++]);

for(ctr = 1 ; ctr <= space2 ; printf(" "), ctr++);

printf("%c", str[end--]);

}

for(ctr = 1 ; ctr <space1 ; printf(" "), ctr++);

printf(" %c",str[start++]);

space2 = 1;

space1 = (len/2)-1;

start--;

printf("\n");

for(row = 1 ; row < (len/2+1); row++, printf("\n"), space1-- , space2+=2)

{

for(ctr = 1 ; ctr <= space1 ; printf(" "), ctr++);

printf("%c",str[--start]);

for(ctr = 1 ; ctr <= space2 ; printf(" "), ctr++);

printf("%c", str[++end]);

}

return 0;

}

**Testcase 1:**

**Input:**

Welcome

**Output:**

w e

e m

l o

c

l o

e m

w e

**Testcase 2:**

**Input:**

hai

**Output:**

h i

a

h i

**Testcase 3:**

**Input:**

Printthepattern

**Output:**

p n

r r

i e

n t

t t

t a

h p

e

h p

t a

t t

n t

i e

r r

p n

**Testcase 4:**

**Input:**

hello

**Output:**

h o

e l

l

e l

h o

**Testcase 5:**

**Input:**

wonderful

**Output:**

w l

o u

n f

d r

e

d r

n f

o u

w l

**Testcase 6:**

**Input:**

cleartheworld

**Output:**

c d

l l

e r

a o

r w

t e

h

t e

r w

a o

e r

l l

c d

**Testcase 7:**

**Input:**

forgive

**Output:**

f e

o v

r i

g

r i

o v

f e

**Testcase 8:**

**Input:**

examlyy

**Output:**

e y

x y

a l

m

a l

x y

e y

**Testcase 9:**

**Input:**

zohoround

**Output:**

z d

o n

h u

o o

r

o o

h u

o n

z d

**Program 18:**

**Sub Topic:** Strings

***Remove Characters***

*﻿*Given two Strings s1 and s2, remove all the characters from s1 which is present in s2.

**Constraints**

1<= string length <= 200

**Sample Input 1**

experience

enc

**Sample Output 1**

xpri

**Solution:**

#include<stdio.h>

int main()

{

char s1[200] , s2[200];

int ind1 , ind2;

scanf("%s%s" , s1 , s2);

for(ind2 = 0 ; s2[ind2] ; ind2++)

{

for(ind1 = 0 ; s1[ind1] ; ind1++)

{

if(s1[ind1] == s2[ind2])

s1[ind1] = '@';

}

}

for(ind1 = 0 ; s1[ind1] && s1[ind1] != '@' ; ind1++);

ind2 = ind1;

while(1)

{

for(ind2; s1[ind2] && s1[ind2] == '@' ; ind2++);

while( s1[ind2] && s1[ind2] != '@')

{

s1[ind1] = s1[ind2];

ind1++;

ind2++;

}

if(s1[ind2] == 0) break;

}

s1[ind1] = 0;

printf("%s" , s1);

return 0;

}

**Testcase 1:**

**Input:**

experience

enc

**Output:**

Xpri

**Testcase 2:**

**Input:**

Fiih+!\,ln:nRZfHJZd'<hFRc

Fiih+!\,ln

**Output:**

:RZfHJZd'<Rc

**Testcase 3:**

**Input:**

Fiih+!\,ln:nRZfHJZd'<hFRcuBvXoQThprY n;v7oOsolrX!e6W9f9;F1PT?08FWpUa)zCrD!)SSaao"MUURDF(o%f>fd:\ ztC

Fiih+!\,ln:nRZfHJZd'<hFRcuBvXo

**Output:**

iih+,ZHJZ'<hcuBQhY

**Testcase 4:**

**Input:**

sdjkhfjdbvjfbvjfbhvjhbfvhjbjbvjhbfrvhjbrfgbvrvhjhjbjbfjbvhjbfhvjbdfbvjbfvhjbfvhjf

bvf

**Output:**

Sdjkhjdjjhjhhjjjhrhjrgrhjhjjjhjhjdjhjhj

**Testcase 5:**

**Input:**

jfbhvjbfjfbvdjlakjdklafckajbvlasndwklfjkleghbvkjbsbckjlehnvkjrvjebfkbvkjbfkje

ak

**Output:**

Jfbhvjbfjfbvdjljdlfcjbvlsndwlfjleghbvjbsbcjlehnvjrvjebfbvjbfje

**Testcase 6:**

**Input:**

vhbfhvjbhfvhjbfvjbfbvjd

dbchfn

**Output:**

Vvjvjvjvj

**Testcase 7:**

**Input:**

cjhbjdbvkjdbvjkeakfbgejugfyuiegfesugfiu

hdhjfbf

**Output:**

Cvkvkeakgeugyuiegesugiu

**Testcase 8:**

**Input:**

dnbvjhbv

dfhvbhjbvb

**Output:**

n

**Testcase 9:**

**Input:**

fdkjvhjkfbvkjbfkjbvfjkbv

abchdb

**Output:**

Fkjvjkfvkjfkjvfjkv

**Program 19:**

**Sub Topic:**Strings

***Remove palindrome words***

string should contains only the words are not palindrome

**Sample Input 1**

Malayalam is my mother tongue

**Sample Output 1**

is my mother tongue

**Solution**

#include<stdio.h>

#include<stdlib.h>

int mystrlen(char \*str)

{

int ind;

for(ind = 0 ; str[ind] ; ind++);

return ind;

}

int isPalindrome(char \*str)

{

int start , end;

for(start = 0 , end = mystrlen(str) - 1 ; start < end ; start++ , end--)

{

if(abs(str[start] - str[end]) == 32 )

continue;

if( str[start] != str[end] )

return 0;

}

return 1;

}

int main()

{

char str[100];

char \*start;

int ind;

scanf("%[^\n]s" , str);

start = str;

for(ind = 0 ; str[ind] ; ind++)

{

if( str[ind] == 32)

{

str[ind] = 0;

if( isPalindrome(start) == 0)

printf("%s " , start);

str[ind] = 32;

start = str + ind + 1;

}

}

if( isPalindrome(start) == 0)

printf("%s " , start);

return 0;

}

**Testcase 1:**

**Input:**

Malayalam is my mother tongue

**Output:**

is my mother tongue

**Testcase 2:**

**Input:**

He did a good deed

**Output:**

He good

**Testcase 3:**

**Input:**

Hah lovevol you

**Output:**

You

**Testcase 4:**

**Input:**

hjds abcba abcd ekdjcb

**Output:**

hjds abcd ekdjcb

**Testcase 5:**

**Input:**

mind Blowolb

**Output:**

Mind

**Testcase 6:**

**Input:**

sdhfdh sdhdges dhfesjhfgesj

**Output:**

sdhfdh sdhdges dhfesjhfgesj

**Testcase 7:**

**Input:**

wonder of youoy

**Output:**

wonder of

**Testcase 8:**

**Input:**

Sdhdshgdvcdhcvdhcsjc

**Output:**

Sdhdshgdvcdhcvdhcsjc

**Testcase 9:**

**Input:**

aaaaabbbbaaaaa hsdcjkdcbjdskcbkjbd

**Output:**

Hsdcjkdcbjdskcbkjbd

**Testcase 10:**

**Input:**

zdvcdhcvdbvjdbvjzdvcdhcvdbvjdbvj cdc fdvhjvbdcvbhdsv

**Output:**

zdvcdhcvdbvjdbvjzdvcdhcvdbvjdbvj fdvhjvbdcvbhdsv

**Program 20**

**Sub Topic:**Strings

***Remove unbalanced parenthesis***

  Remove unbalanced parentheses in a given expression.

**Sample Input 1**

((abc)((de))

**Sample Output 1**

(abc)((de))

**Sample Input 2**

(((ab)

**Sample Output 2**

(ab)

**Solution:**

#include<stdio.h>

#include<malloc.h>

int main()

{

char str[50];

scanf("%s" , str);

int \*arr , \*close , ctop ,len , top , ind , safe , hash , nonhash;

for(len = 0 ; str[len] ; len++);

arr = (int\*)calloc(len , sizeof(int));

close = (int\*)calloc(len , sizeof(int));

ctop = top = -1;

safe = 1;

for(ind = 0 ; str[ind] ; ind++)

{

if(str[ind] == '(')

arr[++top] = ind ;

else if(str[ind] == ')')

{

if( top == -1)

{

close[++ctop] = ind;

continue;

}

arr[top--] = -1;

}

}

for(ind = 0 ; top >= 0 && ind <= top ; ind++)

str[arr[ind]] = '#';

for(ind = 0 ; ctop >= 0 && ind <= ctop ; ind++)

str[close[ind]] = '#';

hash = nonhash = 0;

while(str[nonhash] && str[nonhash] == '#')

{

if(str[++nonhash] != '#')

break;

}

while(str[nonhash])

{

while(str[nonhash] && str[nonhash] == '#')

nonhash++;

str[hash] = str[nonhash];

hash++;

nonhash++;

}

str[hash] = 0;

printf("%s" , str);

return 0;

}

**Testcase 1:**

**Input:**

((abc)((de))

**Output:**

(abc)((de))

**Testcase 2:**

**Input:**

(((ab)

**Output:**

(ab)

**Testcase 3:**

**Input:**

((ab)(c+d))

**Output:**

((ab)(c+d))

**Testcase 4:**

**Input:**

((ab)(cd)(((de)

**Output:**

(ab)(cd)(de)

**Testcase 5:**

**Input:**

((((((((((((((((((ab)

**Output:**

(ab)

**Testcase 6:**

**Input:**

(ab))))))))))

**Output:**

(ab)

**Testcase 7:**

**Input:**

(((((((((ab)((((((cd)(ef)

**Output:**

(ab)(cd)(ef)

**Testcase 8:**

**Input:**

(ab)))))(cd))))))(ef)

**Output:**

(ab)(cd)(ef)

**Testcase 9:**

**Input:**

(ab)))))))))(cd))

**Output:**

(ab)(cd)

**Program 21:**

**Sub Topic:** String

***Reverse a String***

You’re given a string as an input. You have to reverse the string by keeping the punctuation and spaces. You have to modify the source string itself without creating an another string.

**Constraints**

1<=string length<=500

**Sample Input 1**

A man, in the boat says : I see 1-2-3 in the sky

**Sample Output 1**

y kse, ht ni3 21ee sIsy : a sta o-b-e ht nin amA

**Solution:**

#include<stdio.h>

#define isAllow(ch) (ch >='0' && ch <= '9' || ch >= 'a' && ch <= 'z' || ch >= 'A' && ch <= 'Z' )

int main()

{

char str[200] , temp;

scanf("%[^\n]s" , str);

int start , end ;

for(end = 0 ; str[end] ; end++);

end--;

start = 0;

while(start < end)

{

while( str[start] && !(isAllow(str[start]))) start++;

while( str[end] && !(isAllow(str[end]))) end--;

if(start < end)

{

temp = str[start];

str[start] = str[end];

str[end] = temp;

start++;

end--;

}

}

printf("%s " , str);

return 0;

}

**Testcase 1:**

**Input:**

A man, in the boat says : I see 1-2-3 in the sky

**Output:**

y kse, ht ni3 21ee sIsy : a sta o-b-e ht nin amA

**Testcase 2:**

**Input:**

Fiih+!\,ln

**Output:**

nlhi+!\,iF

**Testcase 3:**

**Input:**

Fiih+!\,ln:nRZfHJZd'<hFRcuBvXoQThprY n;v7oOsolrX!e

**Output:**

eXrl+!\,os:Oo7vnYrp'<hTQoXvBucRFhdZJ H;fZRnnlhii!F

**Testcase 4:**

**Input:**

Fiih+!\,ln:nRZfHJZd'<hFRcuBvXoQThprY n;v7oOsolrX!e6W9f9;F1PT?08FWpUa)z

**Output:**

zaUp+!\,WF:80TP1F9f'<9W6eXrlosOo7vnY r;phTQoXvBu!cRFhdZ;JHfZ?Rnnlhii)F

**Testcase 5:**

**Input:**

Fiih+!\,ln:nRZfHJZd'

**Output:**

dZJH+!\,fZ:RnnlhiiF'

**Testcase 6:**

**Input:**

Fiih+!\,ln:nRZfHJZd'<hFRcuBvXoQThprY n;v7oOsolrX!e6W9f9;F1PT?08FWpUa)zCrD!)SSaao"MUURDF(o%f>fd:\ ztC\_mT3T4<7KS5#0@.8onqmICaO6;agk0a%<ki eDh@N<th2B?@?P

**Output:**

PB2h+!\,tN:hDeika0k'<ga6OaCImqno805S K;74T3TmCtz!dffoFD;RUUM?oaaSSDr)CzaU!)pWF80"TP1F9f(9%W>6e:\ Xrl\_osOo7<vnYr#p@.hTQoXvBucR;FhdZJ%<Hf ZRn@n<lhii?@?F

**Testcase 7:**

**Input:**

Fiih+!\,ln:nRZfHJZd'<hFRcuBvXo

**Output:**

oXvB+!\,uc:RFhdZJHf'<ZRnnlhiiF

**Testcase 8:**

**Input:**

Fiih+!\,ln:nRZfHJZd'<hFRcuBvXoQThprY n;v7oOsolrX!e6W9f9;F1PT?08FWpUa)zCrD!)SSaao"MUURDF(o%f>fd:\ ztC\_mT3T4<7KS5#0@.8onqmICaO6;agk0a%<ki eDh@N<th2B?@?PdsI0x5!4<+\_<bQFjk

**Output:**

kjFQ+!\,b4:5x0IsdPB'<2htNhDeika0kga6 O;aCImqno80!5SK74T;3TmC?tzdffoF)DRUU!)MoaaS"SDrCza(U%p>WF:\ 80T\_P1F9f<9W6e#X@.rlosOo7vnY;rphTQ%<oX vBu@c<RFhd?@?ZJHfZRn!n<+\_<lhiiF

**Testcase 9:**

**Input:**

Fiih+!\,ln:nRZfHJZd'<hFRcuBvXoQThprY n;v7oOsolrX!e6W9f9

**Output:**

9f9W+!\,6e:XrlosOo7'<vnYrphTQoXvBucR F;hdZJHfZRn!nlhiiF

**Testcase 10:**

**Input:**

Fiih+!\,ln:nRZfHJZd'<hFRc

**Output:**

cRFh+!\,dZ:JHfZRnnl'<hiiF

**Program 22:**

**Sub Topic:** Strings

***Reverse and Add until get a palindrome***

 Take a number, reverse it and add it to the original number until the obtained number is a palindrome

**Constraints**

1<=num<=99999999

**Sample Input 1**

32

**Sample Output 1**

55

**Solution:**

#include<stdio.h>

long long int reverse(long long int num)

{

long long int revnum = 0;

while(num)

{

revnum = revnum \* 10 +(num%10);

num /= 10;

}

return revnum;

}

int main()

{

long long int num , sum ;

scanf("%lld" , &num);

while(1)

{

sum = num + reverse(num);

if( sum == reverse(sum))

break;

num = sum;

}

printf("%lld" , sum);

return 0;

}

**Testcase 1:**

**Input:**

32

**Output:**

55

**Testcase 2:**

**Input:**

1234

**Output:**

5555

**Testcase 3:**

**Input:**

354656

**Output:**

11244211

**Testcase 4:**

**Input:**

3656

**Output:**

125521

**Testcase 5:**

**Input:**

24546

**Output:**

39633693

**Testcase 6:**

**Input:**

1234567

**Output:**

8888888

**Testcase 7:**

**Input:**

3545

**Output:**

8998

**Testcase 8:**

**Input:**

23

**Output:**

55

**Testcase 9:**

**Input:**

5670

**Output:**

59895

**Testcase 10:**

**Input:**

34

**Output:**

77

**Program 23:**

**Sub Topic:**Arrays

***Sorting Based on no of factors***

To find the factors of the numbers given in an array and to sort the numbers in descending order according to the factors present in it

**Input Format**

Input contains the array size and the values

**Output Format**

print the array which is sorted by the factors count

**Constraints**

1 <= array\_size <= 1000

**Sample Input 1**

5

8 2 3 12 16

**Sample Output 1**

12 16 8 2 3

**Solution:**

#include<stdio.h>

#include<math.h>

int findFactCount( int n)

{

int ctr,sqr,count=2;

sqr = (int)sqrt(n);

if(n == 1)

count--;

else

{

for( ctr = 2 ; ctr<= sqr ; ctr++ )

if( n % ctr == 0 )

count +=2;

ctr--;

if( ctr \* ctr == n )

count--;

}

return count;

}

int main()

{

int arr[1000],fact[1000],ctr,size,max=-1,maxInd=-1,temp,ctr1;

scanf("%d",&size);

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

scanf("%d",&arr[ctr]);

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

fact[ctr]=findFactCount(arr[ctr]);

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

{

for( ctr1 =0 ; ctr1 < size ; ctr1++ )

{

if( fact[ctr1] != -1 && max < fact[ctr1])

{

max = fact[ctr1];

maxInd = ctr1;

}

}

/\* temp = arr[ctr];

arr[ctr] = arr[maxInd];

arr[maxInd] = temp;

temp = fact[ctr];

fact[ctr] = fact[maxInd];

fact[maxInd] = temp;\*/

fact[maxInd] = -1;

printf("%d ",arr[maxInd]);

max = -1;

maxInd=-1;

}

/\*for( ctr =0 ; ctr < size ; ctr++ )

printf("%d ",arr[ctr]);

\*/

}

**Testcase 1:**

**Input:**

5

8 2 3 12 16

**Output:**

12 16 8 2 3

**Testcase 2:**

**Input:**

49

823 923 227 156 105 899 833 708 350 184 833 827 685 781 404 530 927 664 420 470 668 117 131 794 156 248 775 75 164 183 148 632 251 327 305 920 244 693 719 864 606 832 574 756 985 652 721 101 593

**Output:**

420 864 756 920 832 156 708 350 156 693 105 184 530 664 470 248 632 606 574 833 833 404 927 668 117 775 75 164 148 244 652 923 899 685 781 794 183 327 305 985 721 823 227 827 131 251 719 101 593

**Testcase 3**

**Input:**

109

791 206 106 248 137 527 792 401 398 305 155 927 716 273 989 904 858 999 454 220 966 75 845 928 135 541 521 153 495 717 734 559 676 29 629 301 503 994 331 349 637 906 934 920 325 955 364 574 586 424 941 635 298 481 394 659 455 331 823 599 261 957 302 624 668 293 613 155 847 506 26 325 981 939 484 569 639 616 468 8 338 543 293 183 725 569 968 431 437 725 52 201 208 725 557 8 17 864 811 99 491 447 925 266 999 916 638 377 505

**Output:**

792 864 624 468 858 966 920 616 220 928 495 364 968 208 676 484 248 273 904 999 135 994 906 574 424 455 957 506 266 999 638 927 716 75 845 153 637 325 261 668 847 325 981 639 338 725 725 52 725 99 925 916 791 206 106 527 398 305 155 989 454 717 734 559 629 301 934 955 586 635 298 481 394 302 155 26 939 8 543 183 437 201 8 447 377 505 137 401 541 521 29 503 331 349 941 659 331 823 599 293 613 569 293 569 431 557 17 811 491

**Testcase 4:**

**Input:**

586

115 611 259 490 612 624 519 479 47 436 884 366 660 343 885 850 470 453 579 758 63 913 880 219 114 497 177 675 345 937 538 77 120 883 886 643 697 934 199 674 882 405 754 960 91 811 551 275 821 732 447 112 879 886 158 290 185 912 203 149 862 90 765 187 254 342 41 989 626 879 945 500 395 160 758 57 492 694 825 178 87 633 478 784 742 536 235 550 10 588 607 363 137 311 85 930 491 613 195 440 493 433 478 406 946 212 392 905 398 298 972 712 893 438 233 164 971 537 957 429 749 846 820 891 544 520 506 523 568 398 764 563 844 270 541 273 727 153 791 567 561 855 48 226 391 739 912 925 104 256 967 560 162 781 915 126 213 29 362 221 125 411 502 744 437 222 671 18 834 423 500 301 485 291 132 347 174 902 121 699 125 294 695 847 73 337 140 580 718 847 712 18 635 443 870 253 482 815 659 996 355 703 702 641 818 464 25 433 824 377 879 165 0 105 830 930 443 770 382 420 85 677 170 640 864 599 903 251 937 739 90 366 515 520 990 316 961 389 854 814 831 924 72 893 610 379 315 416 990 345 938 863 89 616 840 553 182 526 878 946 676 913 803 487 136 947 828 638 528 547 4 88 209 77 765 337 123 428 655 186 571 173 207 86 417 990 620 955 324 416 878 505 322 114 663 224 107 831 855 270 584 41 882 592 112 454 893 306 852 563 73 88 643 911 302 775 743 754 195 550 321 542 461 146 947 581 94 177 381 636 351 302 642 207 921 629 219 446 801 960 91 488 890 242 188 757 670 347 954 321 744 780 575 950 523 739 303 18 471 218 44 66 6 868 196 310 577 777 881 294 8 347 353 67 377 968 96 457 472 729 618 269 232 355 781 539 137 509 126 753 267 419 714 674 377 761 192 5 329 346 212 244 907 866 367 629 903 394 781 603 972 275 591 637 654 421 3 442 783 32 904 811 441 355 909 787 203 893 788 644 682 385 923 395 761 38 539 661 20 543 29 794 841 223 569 625 24 928 904 645 348 772 501 911 210 108 679 813 858 528 594 275 627 678 94 848 547 40 31 347 375 784 223 710 392 630 840 825 382 261 678 443 57 312 837 702 117 260 671 473 358 49 200 875 21 281 869 557 345 94 682 799 873 149 274 575 112 440 301 62 793 843 421 676 845 590 440 991 427 363 901 494 128 961 648 811 135 450 224 17 642 848 316 290 673 478 232 926 362 717 598 88 577 68 755 961 521 428 895 282 196 707 680 684 231 888 193 683 338 944 967 500 799 536 204 788 496 274 976 609 909 412 596 45 845 830 441 529 301 235 696 7 271 83 922 932

**Output:**

840 840 960 960 660 420 864 990 924 990 990 780 630 624 880 912 912 560 528 528 648 612 882 588 972 828 882 972 450 684 120 945 930 440 520 270 744 870 702 930 770 640 520 616 270 744 714 210 858 594 312 702 440 440 680 888 696 784 324 784 192 490 884 850 675 732 90 765 342 500 160 492 825 550 392 846 820 544 855 126 500 132 294 140 580 996 90 72 315 416 765 620 416 224 855 306 852 550 636 954 950 868 294 968 96 126 644 928 348 108 392 825 260 200 224 500 204 405 112 891 567 48 162 464 592 112 848 112 848 944 496 976 256 676 196 441 676 196 441 366 885 470 114 345 754 290 742 536 195 406 946 712 438 957 429 506 568 273 561 104 915 222 834 174 902 712 824 165 105 830 170 903 366 854 814 610 345 938 182 946 136 638 88 186 322 114 663 584 88 754 195 351 642 488 890 670 66 310 777 472 618 232 903 654 442 783 904 682 385 24 904 645 627 678 40 375 710 678 837 875 345 682 590 494 128 135 642 290 232 598 88 282 231 536 609 830 729 436 63 275 363 212 164 764 844 153 925 18 423 847 847 18 316 428 207 775 207 801 242 188 575 18 44 539 212 244 603 275 637 32 909 788 539 20 772 275 261 117 873 575 845 363 316 68 428 338 788 909 412 596 45 845 932 625 115 611 259 519 343 453 579 758 913 219 497 177 538 77 886 697 934 674 91 551 447 879 886 158 185 203 862 187 254 989 626 879 395 758 57 694 178 87 633 478 235 10 85 493 478 905 398 298 893 537 749 398 791 226 391 781 213 362 221 125 411 502 437 671 301 485 291 699 125 695 718 635 253 482 815 355 703 818 377 879 382 85 515 831 893 553 526 878 913 803 209 77 123 655 86 417 955 878 505 831 454 893 302 321 542 146 581 94 177 381 302 921 629 219 446 91 321 303 471 218 6 8 377 355 781 753 267 674 377 329 346 866 629 394 781 591 355 203 893 923 395 38 543 794 501 679 813 94 382 57 671 473 358 21 869 94 799 274 301 62 793 843 427 901 478 926 362 717 755 895 707 799 274 301 235 922 121 25 961 4 841 49 961 961 529 479 47 937 883 643 199 811 821 149 41 607 137 311 491 613 433 233 971 523 563 541 727 739 967 29 347 73 337 443 659 641 433 0 443 677 599 251 937 739 389 379 863 89 487 947 547 337 571 173 107 41 563 73 643 911 743 461 947 757 347 523 739 577 881 347 353 67 457 269 137 509 419 761 5 907 367 421 3 811 787 761 661 29 223 569 911 547 31 347 223 443 281 557 149 421 991 811 17 673 577 521 193 683 967 7 271 83

**Testcase 5:**

**Input:**

952

340 165 564 289 494 443 644 120 859 506 722 473 193 881 396 660 312 320 732 827 916 806 542 527 618 938 613 723 897 87 255 772 651 716 325 472 419 499 24 429 889 25 696 523 907 95 319 566 694 479 517 614 557 833 116 56 777 111 503 225 851 766 294 432 45 32 886 388 375 471 510 471 952 330 188 107 834 350 857 546 467 390 314 883 225 796 473 938 462 747 50 191 805 916 242 539 795 832 509 807 993 350 742 663 960 559 589 238 998 727 764 634 528 645 682 634 144 438 747 862 289 238 255 254 760 983 801 550 581 189 211 497 90 634 193 964 699 198 739 326 374 578 390 405 109 30 324 563 561 776 979 58 745 935 28 133 507 920 205 160 490 991 688 313 809 553 165 415 895 392 925 474 803 223 385 748 617 144 930 527 748 26 274 72 495 240 34 780 95 950 285 681 715 318 389 343 319 844 796 598 148 869 653 714 308 485 764 326 100 474 171 77 3 908 566 296 842 439 386 843 364 499 241 201 159 834 672 139 811 978 588 644 699 709 874 775 778 158 142 419 247 861 399 694 875 646 857 129 498 477 410 595 409 493 568 353 831 322 651 342 198 165 343 222 963 415 292 776 284 72 433 547 154 33 182 17 943 431 281 515 49 346 116 539 350 365 364 848 146 920 116 359 488 347 227 854 229 540 291 533 995 492 916 311 556 302 304 382 109 728 272 246 21 693 30 804 101 283 324 175 161 418 77 454 408 741 706 67 961 292 704 327 567 722 167 390 598 635 533 589 229 737 861 659 510 288 594 517 827 383 976 901 108 425 525 891 247 763 843 237 494 767 113 829 679 802 898 473 599 121 550 527 720 626 912 865 861 785 695 430 491 609 912 906 858 204 899 559 548 101 838 517 805 109 15 639 35 739 885 94 448 897 785 619 21 67 888 56 986 889 509 759 539 568 324 78 156 373 89 810 880 179 651 370 813 421 426 53 132 46 969 567 513 57 119 145 830 3 551 737 508 728 355 113 211 909 5 154 921 666 198 154 834 924 378 804 311 89 507 296 954 508 356 748 33 513 881 850 632 812 564 418 387 782 771 563 308 16 41 413 946 941 314 801 795 465 947 139 731 919 936 751 45 339 87 339 120 531 149 637 941 444 351 998 529 650 276 109 582 241 657 447 966 172 205 118 866 738 582 8 144 72 307 267 203 795 34 822 63 29 149 106 757 748 473 74 707 54 70 144 880 208 142 625 54 687 855 402 249 343 643 354 566 292 962 83 23 102 134 989 558 281 623 18 408 380 320 168 876 64 874 291 771 276 846 162 994 704 0 179 93 456 554 898 118 746 927 671 815 986 921 174 404 677 765 828 708 963 581 735 858 641 318 566 382 543 259 514 441 21 521 67 648 916 571 207 328 188 209 359 988 680 301 27 114 961 446 688 732 395 848 384 885 97 483 682 562 71 627 521 923 96 48 450 882 373 783 89 131 488 264 476 669 384 704 507 536 535 522 143 646 2 242 729 598 95 201 277 254 965 872 32 146 503 198 334 786 905 644 968 872 639 763 741 778 802 700 119 693 846 832 200 803 921 481 933 277 725 753 154 123 296 208 438 219 867 732 194 913 401 140 45 767 753 828 802 261 606 920 678 785 533 898 259 882 720 887 1 841 121 97 342 393 642 917 980 129 771 160 891 303 412 157 831 439 61 330 4 83 275 828 312 349 53 687 416 587 774 610 951 387 76 715 499 7 777 646 905 266 617 980 523 555 68 881 151 688 381 285 618 239 954 719 157 633 750 919 232 880 893 636 878 346 502 765 976 6 518 107 541 430 155 598 808 989 159 326 603 651 951 229 76 743 827 259 734 220 87 763 890 743 155 11 190 399 663 89 54 381 417 380 528 830 794 408 933 254 286 110 869 944 596 611 275 867 46 407 802 175 650 297 807 416 145 153 840 25 942 278 475 453 994 841 51 545 62 248 7 486 249 912 941 750 492 165 502 444 35 491 434 361 373 658 22 236 647 77 323 649 844 371 377 29 273 234 837 297 303 712 536 612 352 316 585 198 239 236 509 794 418 135 548 891 314 590 807 955 484 327 447 103 993 626 824 710 549 135 694 979 284 303 23 817 403 130 883 823 716 456 441 260 858 868 842 676 646 126 826 188 682 583 333

**Output:**

840 720 720 960 660 780 672 540 924 936 432 528 240 912 912 810 880 880 648 880 528 912 396 588 288 828 450 882 700 828 882 980 828 980 612 120 312 696 510 952 330 546 390 462 760 390 920 930 714 920 728 408 390 510 594 858 888 728 378 120 966 408 168 456 858 680 384 264 384 920 330 312 750 408 750 456 858 144 324 144 324 324 144 144 320 832 704 448 320 704 704 832 340 564 644 732 294 350 350 550 90 198 160 490 392 748 748 72 495 950 308 364 644 342 198 72 350 364 492 693 804 108 525 550 204 156 132 666 198 804 954 748 850 812 564 308 444 650 276 738 72 748 855 558 380 876 276 846 765 708 735 988 732 96 476 522 198 644 968 693 846 200 732 140 342 160 416 774 954 636 765 220 380 650 416 486 492 444 234 352 585 198 260 868 126 405 688 848 304 272 567 976 891 567 208 162 688 848 48 208 891 688 976 944 891 225 225 100 441 484 441 676 165 494 506 806 618 938 897 255 651 472 24 429 56 777 375 834 938 805 795 742 663 238 645 682 438 238 255 189 374 30 561 776 935 165 474 385 285 715 318 598 474 296 834 978 874 861 399 875 646 498 410 595 568 322 651 165 222 776 154 182 488 854 246 30 418 741 598 861 494 861 430 609 906 805 885 897 56 986 759 568 78 651 370 426 969 513 830 154 154 834 296 513 632 418 782 946 795 465 351 582 582 795 822 54 70 54 402 354 962 102 874 994 986 174 318 328 114 885 483 682 627 783 488 536 646 598 872 786 872 741 154 296 438 606 678 642 610 715 777 646 266 555 285 618 232 518 430 598 808 651 890 190 399 663 54 830 286 110 297 942 994 248 165 434 658 273 837 297 712 536 418 135 590 824 710 135 130 646 826 682 64 729 722 916 772 716 325 833 116 45 32 388 188 796 747 50 916 242 539 764 747 801 964 578 28 507 925 844 796 148 764 171 908 775 477 963 292 284 116 539 116 916 556 175 292 722 425 548 639 539 508 909 507 508 356 387 801 45 531 637 657 172 63 292 18 927 404 963 916 207 188 507 242 32 639 725 867 45 261 412 275 387 76 68 603 76 596 275 867 175 153 475 236 844 316 236 548 549 284 716 188 333 16 625 473 542 527 723 87 889 95 319 566 694 517 614 111 851 766 886 471 471 314 473 807 993 559 589 998 634 634 862 254 581 497 634 699 326 979 58 745 133 205 553 415 895 803 527 26 274 34 95 681 343 319 869 485 326 77 566 842 386 843 201 159 699 778 158 142 247 694 129 493 831 343 415 33 943 515 346 365 146 291 533 995 302 382 21 161 77 454 706 327 635 533 589 737 517 901 247 763 843 237 767 679 802 898 473 527 626 865 785 695 899 559 838 517 15 35 94 785 21 889 813 46 57 119 145 551 737 355 921 33 771 413 314 731 339 87 339 998 447 205 118 866 8 267 203 34 106 473 74 707 142 687 249 343 566 134 989 623 291 771 93 554 898 118 746 671 815 921 581 566 382 543 259 514 21 209 301 27 446 395 562 923 669 535 143 95 201 254 965 146 334 905 763 778 802 119 803 921 481 933 753 123 219 194 913 767 753 802 785 533 898 259 393 917 129 771 303 831 687 951 905 381 633 893 878 346 502 6 155 989 159 326 951 259 734 87 763 155 381 417 794 933 254 869 611 46 407 802 807 145 278 453 51 545 62 249 502 35 22 77 323 649 371 377 303 794 314 807 955 327 447 993 626 694 979 303 817 403 842 583 289 25 289 49 961 121 529 961 841 121 4 25 841 361 443 859 193 881 827 613 419 499 523 907 479 557 503 107 857 467 883 191 509 727 983 211 193 739 109 563 991 313 809 223 617 389 653 3 439 499 241 139 811 709 419 857 409 353 433 547 17 431 281 359 347 227 229 311 109 101 283 67 167 229 659 827 383 113 829 599 491 101 109 739 619 67 509 373 89 179 421 53 3 113 211 5 311 89 881 563 41 941 947 139 919 751 149 941 109 241 307 29 149 757 643 83 23 281 0 179 677 641 521 67 571 359 97 71 521 373 89 131 2 277 503 277 401 887 97 157 439 61 83 349 53 587 499 7 617 523 881 151 239 719 157 919 107 541 229 743 827 743 11 89 7 941 491 373 647 29 239 509 103 23 883 823 1

**Testcase 6:**

**Input:**

924

404 289 842 357 808 973 329 774 86 739 883 376 631 517 678 443 458 624 217 922 615 694 105 718 343 146 283 430 411 653 388 614 132 897 689 77 787 989 365 899 979 520 373 3 708 265 986 717 385 550 767 871 507 403 689 925 503 270 615 215 789 241 946 906 699 207 407 26 98 499 683 406 960 109 847 756 237 534 628 230 569 470 639 101 134 876 384 814 642 470 992 890 657 257 210 266 672 315 364 826 756 879 277 510 36 978 545 875 156 707 179 447 116 424 906 659 81 855 543 447 530 9 974 735 829 70 267 621 227 973 263 184 377 547 931 891 897 254 812 912 232 138 240 455 649 572 512 137 690 125 310 354 542 577 760 575 27 784 534 756 364 285 289 345 641 117 148 622 446 882 472 732 65 681 970 574 936 961 40 208 979 670 620 687 109 676 959 545 33 239 842 222 459 938 961 639 292 478 993 238 802 462 148 510 392 930 18 516 265 992 250 319 491 513 665 596 41 235 51 93 929 546 656 195 640 277 744 401 211 255 786 203 535 946 712 431 707 211 288 922 836 418 7 230 369 648 127 91 665 998 459 980 698 787 391 547 251 766 205 48 572 805 638 141 196 406 606 525 643 356 243 663 369 109 887 246 359 33 779 411 757 364 943 973 619 8 24 311 548 685 857 306 920 227 917 605 785 802 444 139 664 935 604 468 729 825 33 518 36 901 120 725 773 551 976 688 574 298 19 982 76 506 316 449 863 119 16 842 750 141 18 158 299 786 882 295 530 448 70 206 808 988 436 591 921 429 902 53 506 481 399 449 753 516 373 61 842 819 887 29 755 392 993 248 674 183 846 35 837 208 725 550 915 229 39 268 857 759 936 5 727 812 212 830 788 125 228 268 663 165 201 269 216 339 904 152 519 436 519 197 699 690 367 567 661 834 498 705 685 117 7 327 845 893 995 871 534 859 954 512 664 966 423 826 694 118 0 926 624 593 254 295 74 134 539 120 254 745 941 784 726 967 610 348 604 218 76 780 861 903 263 545 83 69 40 196 499 521 633 171 615 672 354 0 444 596 426 82 662 411 215 544 514 660 954 171 44 324 806 923 323 488 810 661 458 988 101 732 160 691 794 30 482 960 221 376 200 556 852 940 221 248 497 721 802 134 887 516 38 483 98 366 687 40 124 702 156 809 606 785 654 125 420 727 515 712 972 719 485 77 48 654 867 944 218 212 791 211 360 343 592 703 816 100 400 29 373 819 145 285 650 99 403 207 318 262 369 689 521 950 357 577 887 568 369 387 299 144 663 68 37 693 11 263 383 278 200 584 132 245 992 744 49 579 49 441 169 484 764 387 804 989 623 894 880 35 789 504 987 733 516 769 919 685 664 877 15 747 198 953 617 755 563 946 413 360 982 683 597 480 20 806 780 610 771 917 716 718 328 656 19 732 733 729 718 447 821 63 417 51 26 736 221 133 89 387 139 970 740 141 329 955 289 681 202 823 809 725 722 361 251 779 923 980 73 486 481 956 516 774 853 127 764 653 440 779 634 602 350 634 610 678 805 388 787 486 542 563 121 680 99 717 420 241 714 549 50 694 433 963 683 1 74 605 404 543 713 638 94 328 28 503 95 511 802 803 634 195 599 391 379 564 223 617 997 146 504 672 927 130 974 532 556 938 625 816 226 136 585 661 198 573 573 889 149 208 40 304 942 521 661 865 399 50 877 973 277 869 591 43 344 922 607 272 197 583 305 245 422 206 543 417 504 81 250 336 171 225 574 540 727 911 577 940 517 858 950 474 852 523 364 217 461 317 12 870 40 602 475 78 168 489 948 481 173 564 43 477 545 369 173 470 302 319 683 261 981 132 505 351 681 956 938 511 886 453 742 326 812 697 751 658 321 771 568 509 197 237 959 479 871 338 244 830 113 431 106 870 559 368 783 684 558 520 751 871 675 816 293 649 841 308 4 257 487 491 42 469 633 590 334 367 661 410 68 205 137 673 231 38 396 413 127 793 570 535 963 168 100 783 264 925 591 613 130 612 955 440 186 106 671 309 458 860 948 754 565 186 487 203 2 197 690 236 290 847 941 824 527 738 185 546 664 557

**Output:**

960 960 756 672 756 756 936 936 780 672 660 420 360 504 360 480 780 420 504 672 504 540 624 912 240 648 624 810 816 880 816 336 816 882 288 980 468 882 972 980 684 396 612 520 270 384 210 510 690 760 462 510 930 546 640 744 920 120 750 216 690 966 120 702 744 440 680 714 858 870 168 870 520 570 168 264 440 690 546 784 784 324 400 144 448 774 132 708 550 876 992 315 364 156 855 735 812 572 364 732 620 392 516 992 836 572 525 364 306 444 825 988 516 819 392 846 550 812 228 954 726 348 444 544 954 988 732 160 200 852 940 516 156 819 650 950 693 200 132 992 804 516 198 732 736 740 486 516 774 350 486 564 532 585 198 940 950 852 364 948 564 132 812 558 675 308 860 948 738 891 512 208 656 48 976 688 208 567 512 48 944 592 656 208 304 272 368 36 676 196 36 196 100 441 484 225 100 357 808 376 678 615 105 430 897 986 385 615 946 906 406 534 230 470 814 642 470 890 266 826 978 875 424 906 530 70 621 184 897 232 138 455 310 354 534 285 345 472 970 574 40 670 222 459 938 238 250 513 665 195 255 786 946 712 418 230 665 459 805 638 406 606 663 246 24 664 935 518 574 506 786 530 70 808 429 902 506 399 248 837 915 759 830 663 165 904 152 834 498 705 534 664 826 610 861 903 40 615 354 426 806 488 30 376 248 483 366 40 606 654 712 654 285 318 357 568 663 584 894 987 664 946 806 610 328 970 602 610 678 805 638 328 195 130 938 136 40 942 399 344 250 574 474 40 602 78 470 351 938 742 658 568 830 783 42 590 410 231 783 130 186 754 186 290 824 664 729 729 404 388 507 925 207 98 847 628 639 657 116 931 575 117 148 639 292 148 18 596 369 356 243 369 548 605 604 725 76 316 18 436 725 268 212 788 268 436 117 845 423 539 604 76 171 596 171 44 556 98 124 867 212 99 207 369 369 387 68 245 764 387 747 20 716 63 387 725 722 956 764 388 99 549 50 963 605 404 28 927 556 50 245 171 12 475 477 369 261 981 956 338 244 68 963 925 236 847 81 16 625 81 842 973 329 86 517 458 217 922 694 718 343 146 411 614 689 77 989 365 899 979 265 717 767 871 403 689 215 789 699 407 26 237 134 879 545 707 447 543 447 974 267 973 377 254 649 125 542 27 622 446 65 681 979 687 959 545 33 842 478 993 802 265 319 235 51 93 203 535 707 922 91 998 698 391 766 205 141 33 779 411 943 973 8 685 917 785 802 33 901 551 298 982 119 842 141 158 299 295 206 591 921 481 753 842 755 993 674 183 35 39 125 201 339 519 519 699 685 327 893 995 871 694 118 926 254 295 74 134 254 745 218 545 69 633 82 662 411 215 514 923 323 458 794 482 221 221 497 721 802 134 38 687 785 125 515 485 77 218 791 343 703 145 403 262 689 299 278 579 989 623 35 789 685 15 755 413 982 597 771 917 718 718 447 417 51 26 221 133 141 329 955 681 202 779 923 481 779 634 634 542 717 694 74 543 713 94 95 511 802 803 634 391 146 974 226 573 573 889 865 973 869 591 922 583 305 422 206 543 417 517 217 489 481 545 302 319 505 681 511 886 453 326 697 321 771 237 959 871 106 559 871 649 469 633 334 205 38 413 793 535 591 955 106 671 309 458 565 203 527 185 289 9 289 961 961 49 49 169 289 361 121 841 4 739 883 631 443 283 653 787 373 3 503 241 499 683 109 569 101 257 277 179 659 829 227 263 547 137 577 641 109 239 491 41 929 277 401 211 431 211 7 127 787 547 251 643 109 887 359 757 619 311 857 227 139 773 19 449 863 53 449 373 61 887 29 229 857 5 727 269 197 367 661 7 859 0 593 941 967 263 83 499 521 0 661 101 691 887 809 727 719 211 29 373 521 577 887 37 11 263 383 733 769 919 877 953 617 563 683 19 733 821 89 139 823 809 251 73 853 127 653 787 563 241 433 683 503 599 379 223 617 997 661 149 521 661 877 277 43 607 197 727 911 577 523 461 317 173 43 173 683 751 509 197 479 113 431 751 293 257 487 491 367 661 137 673 127 613 487 2 197 941 557 1

**Testcase 7:**

**Input:**

518

523 586 850 230 763 226 243 33 587 176 142 674 658 366 982 801 222 82 172 787 461 431 93 191 485 955 310 330 953 908 486 750 685 548 563 798 492 255 480 805 887 131 21 304 761 904 271 625 593 695 786 245 760 351 20 976 44 478 75 992 971 821 738 711 20 287 968 829 779 780 498 909 875 681 770 214 361 626 119 30 758 858 255 146 997 26 962 249 70 120 386 1 922 98 344 889 184 260 280 706 127 711 287 96 717 514 588 997 800 12 597 223 587 307 956 945 83 535 802 24 866 931 521 711 12 845 184 909 32 19 339 796 930 832 618 718 854 39 603 459 615 24 90 942 457 337 960 849 355 903 729 222 706 749 458 152 34 339 915 689 230 603 450 917 930 104 336 190 926 992 872 394 158 492 61 58 16 501 803 820 733 257 481 59 132 670 565 574 668 966 504 518 872 76 76 332 224 126 67 530 845 364 378 272 669 120 750 275 112 142 645 193 257 134 337 816 657 815 440 975 481 54 999 749 327 63 377 521 796 140 282 585 259 636 33 551 507 325 709 593 96 629 879 824 855 781 657 907 297 559 845 733 91 266 457 564 467 908 673 948 749 791 462 35 783 615 910 366 617 859 35 380 362 817 752 785 807 479 346 814 572 840 931 279 72 747 781 517 132 746 213 757 309 137 166 160 306 830 80 837 367 649 903 836 964 891 49 60 512 524 619 312 970 847 851 421 281 209 651 455 898 274 476 844 282 106 266 305 452 136 703 67 709 31 34 993 999 987 873 153 723 779 468 848 714 798 612 486 766 306 60 825 612 907 799 249 901 802 410 412 581 110 171 306 947 215 459 484 299 104 328 712 312 52 733 864 70 165 454 252 390 105 79 677 859 622 981 532 446 200 95 258 789 888 573 89 273 814 986 3 199 427 446 674 807 256 31 471 732 212 778 174 907 772 788 890 423 883 750 933 620 452 56 219 675 861 620 864 766 855 503 79 237 467 884 117 319 247 522 541 436 993 678 228 419 188 116 640 896 481 298 445 481 815 645 736 576 214 489 296 323 0 211 377 219 761 297 106 82 753 444 82 656 55 644 535 919 800 765 775 70 928 674 202 667 257 759 79 101 154 51 263 496 321 317 319 226 332 965 690 678 908 351 166 63 732 628 965 751 537 754 135 240 530 157 862 243 849

**Output:**

840 960 480 780 504 864 864 576 336 816 240 588 800 450 468 612 612 252 800 330 750 798 760 770 858 120 280 945 930 930 966 378 120 750 440 462 910 312 714 798 312 390 888 750 640 896 690 832 850 486 492 992 738 968 260 96 90 992 492 820 132 224 126 364 975 140 585 636 96 855 564 948 380 572 72 132 160 306 836 60 476 486 306 60 825 306 532 200 732 620 675 620 855 884 522 228 736 444 644 765 928 732 176 304 976 272 112 752 80 891 512 848 656 496 484 256 230 658 366 222 310 255 805 904 786 351 498 875 30 255 962 70 344 184 24 184 618 854 459 615 24 942 903 222 152 915 230 104 190 872 670 574 518 872 530 645 54 999 282 824 297 266 783 615 366 814 830 837 903 970 651 455 282 266 136 999 987 410 110 459 104 328 712 70 165 105 258 273 814 986 174 890 56 861 678 645 296 297 70 759 154 678 351 754 135 530 729 243 801 172 908 548 245 20 44 75 711 20 909 98 711 12 956 931 711 12 845 909 32 796 603 603 668 76 76 332 845 275 657 63 796 507 325 657 845 908 931 279 747 964 524 847 844 452 873 153 412 171 52 981 212 772 788 423 452 117 436 188 116 775 332 908 63 628 243 625 16 586 763 226 33 142 674 982 82 93 485 955 685 21 695 478 287 779 681 214 626 119 758 146 26 249 386 922 889 706 287 717 514 597 535 802 866 339 718 39 849 355 706 749 458 34 339 689 917 926 394 158 58 501 803 481 565 669 142 134 815 481 749 327 377 259 33 551 629 879 781 559 91 749 791 35 35 362 817 785 807 346 781 517 746 213 309 166 649 851 209 898 274 106 305 703 34 993 723 779 766 799 249 901 802 581 215 299 454 622 446 95 789 573 427 446 674 807 471 778 933 219 766 237 319 247 993 481 298 445 481 815 214 489 323 377 219 106 82 753 82 55 535 674 202 667 51 321 319 226 965 166 965 537 862 849 361 49 523 587 787 461 431 191 953 563 887 131 761 271 593 971 821 829 997 127 997 223 587 307 83 521 19 457 337 61 733 257 59 67 193 257 337 521 709 593 907 733 457 467 673 617 859 479 757 137 367 619 421 281 67 709 31 907 947 733 79 677 859 89 3 199 31 907 883 503 79 467 541 419 0 211 761 919 257 79 101 263 317 751 157 1

**Testcase 8:**

**Input:**

69

658 519 610 570 630 189 299 413 467 770 997 401 595 348 259 338 465 368 616 509 839 147 894 408 696 671 380 187 527 89 72 559 639 449 605 774 173 194 750 564 211 866 132 972 535 148 220 510 483 252 861 327 693 922 848 369 266 668 65 669 983 509 45 62 96 170 884 784 421

**Output:**

630 972 252 570 770 616 408 696 750 510 784 348 380 72 774 564 132 220 693 96 884 368 848 658 610 189 595 465 894 483 861 266 170 338 147 639 605 148 369 668 45 519 299 413 259 671 187 527 559 194 866 535 327 922 65 669 62 467 997 401 509 839 89 449 173 211 983 509 421

**Testcase 9:**

**Input:**

118

737 681 545 340 64 814 900 325 615 353 556 899 104 50 48 782 478 284 485 732 726 130 996 86 65 649 396 233 519 183 116 762 125 988 988 910 681 920 836 409 678 365 84 461 12 879 566 549 737 185 825 642 766 330 552 366 299 472 507 927 533 947 915 974 257 485 195 105 433 513 398 118 46 586 604 129 806 869 201 400 289 577 591 141 839 421 850 475 324 532 460 44 278 53 995 490 814 253 888 421 717 209 162 989 234 670 946 297 433 540 815 399 901 579 453 244 161 252

**Output:**

900 540 396 252 910 920 330 552 888 400 324 340 732 726 996 988 988 836 84 825 850 532 460 490 234 48 162 814 615 104 782 130 762 678 642 366 472 915 195 105 513 806 814 670 946 297 399 64 325 556 50 284 116 12 549 507 927 604 475 44 244 737 681 545 899 478 485 86 65 649 519 183 125 681 365 879 566 737 185 766 299 533 974 485 398 118 46 586 129 869 201 591 141 278 995 253 717 209 989 815 901 579 453 161 289 353 233 409 461 947 257 433 577 839 421 53 421 433

**Testcase 10:**

**Input:**

538

358 356 299 425 112 131 564 675 702 652 916 154 858 264 320 615 446 927 748 543 91 163 109 258 961 750 87 613 168 437 599 712 151 786 632 865 395 956 544 330 586 34 408 377 151 56 87 231 265 920 561 660 680 756 951 57 624 651 310 173 633 931 396 452 573 110 392 76 660 640 218 946 701 864 443 309 774 333 678 929 877 947 630 392 200 997 370 491 42 365 999 999 399 394 581 286 698 770 757 722 644 168 201 26 800 687 812 998 933 128 955 613 300 384 788 202 986 462 598 290 224 987 218 857 657 371 409 958 20 307 214 764 355 940 109 522 683 455 619 629 109 265 176 100 900 480 947 84 184 750 564 861 66 558 112 581 641 429 672 428 316 212 563 528 537 451 182 602 574 991 513 680 750 287 828 492 470 427 25 165 904 950 124 132 583 222 832 66 13 161 251 872 554 843 381 175 134 453 357 650 512 832 453 232 807 66 23 270 618 752 24 868 227 709 524 175 145 66 610 808 478 415 793 146 54 200 323 929 665 723 265 730 195 81 176 965 202 732 106 434 71 696 487 523 273 700 14 582 470 819 991 903 244 598 803 91 951 460 358 615 315 61 931 658 669 555 496 856 125 898 44 201 447 370 39 449 594 851 697 857 703 80 294 511 950 838 247 761 574 671 228 655 540 522 841 930 335 942 543 15 737 279 37 929 248 759 79 870 781 345 141 256 36 626 687 160 46 825 23 662 836 56 527 539 532 955 414 26 249 569 221 620 238 769 446 268 27 791 87 306 163 694 611 716 317 336 114 674 178 36 955 280 230 944 319 353 755 231 223 106 50 172 280 910 491 128 790 8 528 265 319 181 36 131 530 155 281 571 64 477 499 674 793 17 911 374 818 19 873 479 774 315 500 897 714 492 170 69 53 231 310 621 344 912 200 448 483 942 714 258 104 593 897 496 451 791 918 723 250 288 821 387 34 11 245 525 898 808 681 864 380 159 823 476 577 49 727 692 284 894 590 270 114 110 532 289 841 436 106 542 382 744 504 130 131 772 518 251 449 361 865 808 74 2 633 86 907 745 889 936 375 577 702 119 215 485 763 859 193 767 220 560 63 803 98 909 586 84 207 137 830 255 138 454 671 415 661 346 604 449 461 276 660 241 730 188 678 468 224 734 277 816 124 841 684 248 747 715 918 957 552 423 513 194 580 377 100 806 715 561 950 743 724 831 969 581 473 38

**Output:**

900 660 756 660 864 630 480 672 540 864 504 936 660 624 528 336 528 912 560 816 396 800 300 828 700 288 468 684 702 858 264 750 168 330 408 920 680 640 770 168 384 462 750 680 750 270 696 594 930 870 280 280 910 714 714 918 270 744 702 918 552 320 832 832 448 564 675 748 544 392 774 392 200 644 812 224 940 522 84 564 558 492 950 132 650 868 200 732 819 460 315 294 950 228 522 160 825 836 532 414 620 306 774 315 500 492 200 525 380 476 532 220 84 276 224 580 950 112 176 112 512 752 176 496 80 944 496 100 256 36 36 36 100 154 615 258 712 786 632 56 231 561 651 310 110 946 678 370 42 999 999 399 286 128 986 598 290 987 455 184 861 66 429 182 602 574 513 470 165 904 222 66 872 357 232 66 618 24 66 610 808 54 665 730 195 434 273 582 470 903 598 615 658 555 856 370 574 942 248 759 345 56 238 114 230 231 128 790 530 374 897 170 231 310 621 344 483 942 258 104 897 250 808 894 590 114 110 130 518 808 375 830 255 138 730 678 248 715 957 513 806 715 561 969 64 356 425 652 916 927 956 931 452 76 333 722 788 657 20 764 428 316 212 124 175 524 175 244 931 44 279 539 268 716 50 172 477 873 387 245 692 284 436 772 63 98 909 207 604 188 124 747 423 724 81 358 299 446 543 91 87 437 865 395 586 34 377 87 265 951 57 633 573 218 309 365 394 581 698 201 26 687 998 933 955 202 218 371 958 214 355 629 265 581 537 451 287 427 583 161 554 843 381 134 453 453 807 145 478 415 793 146 323 723 265 965 202 106 14 803 91 951 358 669 125 898 201 447 39 851 697 703 511 838 247 671 655 335 543 15 737 781 141 626 687 46 662 527 955 26 249 221 446 27 791 87 694 611 674 178 955 319 755 106 8 265 319 155 674 793 818 69 451 791 723 34 898 681 159 106 542 382 865 74 633 86 745 889 119 215 485 763 767 803 586 454 671 415 346 734 194 377 831 581 473 38 961 25 841 49 289 841 361 841 131 163 109 613 599 151 151 173 701 443 929 877 947 997 491 757 613 857 409 307 109 683 619 109 947 641 563 991 13 251 23 227 709 929 71 487 523 991 61 449 857 761 37 929 79 23 569 769 163 317 353 223 491 181 131 281 571 499 17 911 19 479 53 593 821 11 823 577 727 131 251 449 2 907 577 859 193 137 661 449 461 241 277 743

**Testcase 11:**

**Input:**

36

10 8 17 8 43 12 37 39 12 37 31 1 35 10 11 3 12 33 41 27 48 30 39 43 35 26 46 11 21 38 18 16 1 5 40 35

**Output:**

48 30 40 12 12 12 18 16 10 8 8 39 35 10 33 27 39 35 26 46 21 38 35 17 43 37 37 31 11 3 41 43 11 5 1 1

**Testcase 12:**

**Input:**

32

9 30 11 30 2 28 4 8 43 49 2 17 46 46 9 20 8 32 44 26 3 45 10 7 19 17 39 46 46 29 34 6

**Output:**

30 30 28 20 32 44 45 8 46 46 8 26 10 39 46 46 34 6 9 4 49 9 11 2 43 2 17 3 7 19 17 29

**Program 24**

**Sub Topic:**Algorithms

**Input:**

***Spiral pattern***

**Sample Input 1**

5

**Sample Output 1**

5 5 5 5 5 5 5 5 5

5 4 4 4 4 4 4 4 5

5 4 3 3 3 3 3 4 5

5 4 3 2 2 2 3 4 5

5 4 3 2 1 2 3 4 5

5 4 3 2 2 2 3 4 5

5 4 3 3 3 3 3 4 5

5 4 4 4 4 4 4 4 5

5 5 5 5 5 5 5 5 5

**Solution:**

#include<stdio.h>

int setNum\_Pow(int n , int \*pow )

{

int num = 0 , ind ;

\*pow = 1;

for(ind = 0 ; ind < n ; num = num \* 10 + n , (\*pow) = \*pow \* 10 + 1 , ind++);

(\*pow) /= 100;

return num/10;

}

int reverse(int num)

{

int revnum = 0 ;

while(num)

{

revnum = revnum \* 10 + (num % 10);

num /= 10;

}

return revnum;

}

int main()

{

int N , num , row , count , col , pow , revnum, POW;

scanf("%d" , &N);

num = setNum\_Pow(N-1 , &pow );

for(row= 1 ; row <= (2\*N-1) ; row++)

printf("%d " , N);

printf("\n");

count = N-1;

for(row = 2 ; row <= N ; row++ , printf("\n"))

{

printf("%d " ,N);

POW = 1;

revnum = reverse(num);

while(revnum / POW)

{

printf("%d " , (revnum/POW) % 10);

POW \*= 10;

}

printf("%d " , count--);

POW = 1;

while(num / POW)

{

printf("%d " , (num/POW) % 10);

POW \*= 10;

}

printf("%d " ,N);

pow /= 10;

num = num - pow ;

}

// second half

count = 2 ;

for(row = 2 ; row <= N ; row++ , printf("\n"))

{

printf("%d " ,N);

POW = 1;

revnum = reverse(num);

while(revnum / POW)

{

printf("%d " , (revnum/POW) % 10);

POW \*= 10;

}

printf("%d " , count++);

POW = 1;

while(num / POW)

{

printf("%d " , (num/POW) % 10);

POW \*= 10;

}

printf("%d " ,N);

pow = pow \* 10 + 1;

num = num + pow ;

}

return 0;

}

**Testcase 1:**

**Input:**

5

**Output:**

5 5 5 5 5 5 5 5 5

5 4 4 4 4 4 4 4 5

5 4 3 3 3 3 3 4 5

5 4 3 2 2 2 3 4 5

5 4 3 2 1 2 3 4 5

5 4 3 2 2 2 3 4 5

5 4 3 3 3 3 3 4 5

5 4 4 4 4 4 4 4 5

5 5 5 5 5 5 5 5 5

**Testcase 2:**

**Input:**

3

**Output:**

3 3 3 3 3

3 2 2 2 3

3 2 1 2 3

3 2 2 2 3

3 3 3 3 3

**Testcase 3:**

**Input:**

7

**Output:**

7 7 7 7 7 7 7 7 7 7 7 7 7

7 6 6 6 6 6 6 6 6 6 6 6 7

7 6 5 5 5 5 5 5 5 5 5 6 7

7 6 5 4 4 4 4 4 4 4 5 6 7

7 6 5 4 3 3 3 3 3 4 5 6 7

7 6 5 4 3 2 2 2 3 4 5 6 7

7 6 5 4 3 2 1 2 3 4 5 6 7

7 6 5 4 3 2 2 2 3 4 5 6 7

7 6 5 4 3 3 3 3 3 4 5 6 7

7 6 5 4 4 4 4 4 4 4 5 6 7

7 6 5 5 5 5 5 5 5 5 5 6 7

7 6 6 6 6 6 6 6 6 6 6 6 7

7 7 7 7 7 7 7 7 7 7 7 7 7

**Testcase 4:**

**Input:**

2

**Output:**

2 2 2

2 1 2

2 2 2

**Testcase 5:**

**Input:**

4

**Output:**

4 4 4 4 4 4 4

4 3 3 3 3 3 4

4 3 2 2 2 3 4

4 3 2 1 2 3 4

4 3 2 2 2 3 4

4 3 3 3 3 3 4

4 4 4 4 4 4 4

**Testcase 6:**

**Input:**

9

**Output:**

9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9

9 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 9

9 8 7 7 7 7 7 7 7 7 7 7 7 7 7 8 9

9 8 7 6 6 6 6 6 6 6 6 6 6 6 7 8 9

9 8 7 6 5 5 5 5 5 5 5 5 5 6 7 8 9

9 8 7 6 5 4 4 4 4 4 4 4 5 6 7 8 9

9 8 7 6 5 4 3 3 3 3 3 4 5 6 7 8 9

9 8 7 6 5 4 3 2 2 2 3 4 5 6 7 8 9

9 8 7 6 5 4 3 2 1 2 3 4 5 6 7 8 9

9 8 7 6 5 4 3 2 2 2 3 4 5 6 7 8 9

9 8 7 6 5 4 3 3 3 3 3 4 5 6 7 8 9

9 8 7 6 5 4 4 4 4 4 4 4 5 6 7 8 9

9 8 7 6 5 5 5 5 5 5 5 5 5 6 7 8 9

9 8 7 6 6 6 6 6 6 6 6 6 6 6 7 8 9

9 8 7 7 7 7 7 7 7 7 7 7 7 7 7 8 9

9 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 9

9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9

**Testcase 7:**

**Input:**

8

**Output:**

8 8 8 8 8 8 8 8 8 8 8 8 8 8 8

8 7 7 7 7 7 7 7 7 7 7 7 7 7 8

8 7 6 6 6 6 6 6 6 6 6 6 6 7 8

8 7 6 5 5 5 5 5 5 5 5 5 6 7 8

8 7 6 5 4 4 4 4 4 4 4 5 6 7 8

8 7 6 5 4 3 3 3 3 3 4 5 6 7 8

8 7 6 5 4 3 2 2 2 3 4 5 6 7 8

8 7 6 5 4 3 2 1 2 3 4 5 6 7 8

8 7 6 5 4 3 2 2 2 3 4 5 6 7 8

8 7 6 5 4 3 3 3 3 3 4 5 6 7 8

8 7 6 5 4 4 4 4 4 4 4 5 6 7 8

8 7 6 5 5 5 5 5 5 5 5 5 6 7 8

8 7 6 6 6 6 6 6 6 6 6 6 6 7 8

8 7 7 7 7 7 7 7 7 7 7 7 7 7 8

8 8 8 8 8 8 8 8 8 8 8 8 8 8 8

**Testcase 8:**

**Input:**

6

**Output:**

6 6 6 6 6 6 6 6 6 6 6

6 5 5 5 5 5 5 5 5 5 6

6 5 4 4 4 4 4 4 4 5 6

6 5 4 3 3 3 3 3 4 5 6

6 5 4 3 2 2 2 3 4 5 6

6 5 4 3 2 1 2 3 4 5 6

6 5 4 3 2 2 2 3 4 5 6

6 5 4 3 3 3 3 3 4 5 6

6 5 4 4 4 4 4 4 4 5 6

6 5 5 5 5 5 5 5 5 5 6

6 6 6 6 6 6 6 6 6 6 6

**Program 25**

**Sub Topic:**Strings

***String pattern***

print the string as the following pattern

(only for odd length string)

**Sample Input 1**

Hello

**Sample Output 1**

l

ll

llo

lloH

lloHe

**Solution:**

#include<stdio.h>

int main()

{

char str[100];

scanf("%s" , str);

int len ,row , col , mid , space;

for(len = 0 ; str[len] ; len++);

space = len - 1;

for(row = 1 ; row <= len/2+1 ; row++,space-- ,printf("\n"))

{

for(col = 1 ; col <= space ; printf(" "), col++);

for(col = 1 , mid = len/2 ; col <= (len - space) ; col++)

printf("%c", str[mid++]);

}

//space+=2;

for(row = 1 ; row <= len/2 ; row++,space-- ,printf("\n"))

{

for(col = 1 ; col <= space ; printf(" "), col++);

for(mid = len/2 ; str[mid]; mid++)

printf("%c", str[mid]);

for(col = 1 ; col <= row ; col++)

printf("%c" , str[col-1]);

}

return 0;

}

**Testcase 1:**

**Input:**

Hello

**Output:**

l

ll

llo

lloH

lloHe

**Testcase 2:**

**Input:**

PROGRAM

**Output:**

G

GR

GRA

GRAM

GRAMP

GRAMPR

GRAMPRO

**Testcase 3:**

**Input:**

WONDER

**Output:**

D

DE

DER

DERS

DERSW

DERSWO

DERSWON

**Testcase 4:**

**Input:**

LOVELYO

**Output:**

E

EL

ELY

ELYO

ELYOL

ELYOLO

ELYOLOV

**Testcase 5:**

**Input:**

12345

**Output:**

3

34

345

3451

34512

**Testcase 6:**

**Input:**

ONETWOTHREE

**Output:**

O

OT

OTH

OTHR

OTHRE

OTHREE

OTHREEO

OTHREEON

OTHREEONE

OTHREEONET

OTHREEONETW

**Testcase 7:**

**Input:**

TWO

**Output:**

W

WO

WOT

**Testcase 8:**

**Input:**

ELEVENN

**Output:**

V

VE

VEN

VENN

VENNE

VENNEL

VENNELE

**Testcase 9:**

**Input:**

LOVEINDIA

**Output:**

I

IN

IND

INDI

INDIA

INDIAL

INDIALO

INDIALOV

INDIALOVE

**Testcase 10:**

**Input:**

ABCDEFGHI

**Output:**

E

EF

EFG

EFGH

EFGHI

EFGHIA

EFGHIAB

EFGHIABC

EFGHIABCD

**Program 26**

**Sub Topic:**Algorithms

***Triangle Pattern***

***Note : Don't use any matrix***

***N= 7***

1 8 14 19 23 26 28

2 9 15 20 24 27

3 10 16 21 25

4 11 17 22

5 12 18

6 13

7

**Sample Input 1**

5

**Sample Output 1**

1 6 10 13 15

2 7 11 14

3 8 12

4 9

5

**Solution:**

#include<stdio.h>

int main()

{

int N , row ,col , val , counter;

scanf("%d" , &N);

for(row = N ; row > 0 ; row-- , printf("\n"))

{

printf("%d ", N-row+1);

for(counter = N-1 ,val = 1 ,val = val + counter + (N - row+1 ), col = 1 ; col < row ; col++ , counter--)

{

printf("%d " , val);

val += counter;

}

}

return 0;

}

**Testcase 1:**

**Input:**

5

**Output:**

1 6 10 13 15

2 7 11 14

3 8 12

4 9

5

**Testcase 2:**

**Input:**

7

**Output:**

1 8 14 19 23 26 28

2 9 15 20 24 27

3 10 16 21 25

4 11 17 22

5 12 18

6 13

7

**Testcase 3:**

**Input:**

8

**Output:**

1 9 16 22 27 31 34 36

2 10 17 23 28 32 35

3 11 18 24 29 33

4 12 19 25 30

5 13 20 26

6 14 21

7 15

8

**Testcase 4:**

**Input:**

10

**Output:**

1 11 20 28 35 41 46 50 53 55

2 12 21 29 36 42 47 51 54

3 13 22 30 37 43 48 52

4 14 23 31 38 44 49

5 15 24 32 39 45

6 16 25 33 40

7 17 26 34

8 18 27

9 19

10

**Testcase 5:**

**Input:**

4

**Output:**

1 5 8 10

2 6 9

3 7

4

**Testcase 6**

**Input:**

6

**Output:**

1 7 12 16 19 21

2 8 13 17 20

3 9 14 18

4 10 15

5 11

6

**Testcase 7:**

**Input:**

2

**Output:**

1 3

2

**Testcase 8:**

**Input:**

9

**Output:**

1 10 18 25 31 36 40 43 45

2 11 19 26 32 37 41 44

3 12 20 27 33 38 42

4 13 21 28 34 39

5 14 22 29 35

6 15 23 30

7 16 24

8 17

9

**Testcase 9:**

**Input:**

14

**Output:**

1 15 28 40 51 61 70 78 85 91 96 100 103 105

2 16 29 41 52 62 71 79 86 92 97 101 104

3 17 30 42 53 63 72 80 87 93 98 102

4 18 31 43 54 64 73 81 88 94 99

5 19 32 44 55 65 74 82 89 95

6 20 33 45 56 66 75 83 90

7 21 34 46 57 67 76 84

8 22 35 47 58 68 77

9 23 36 48 59 69

10 24 37 49 60

11 25 38 50

12 26 39

13 27

14

**Program 27**

**Sub topic:** Strings

***WORD REVERSAL USING RECURSIVE***

Using Recursion to reverse the string such as

**Input Format**

one two three

**Output Format**

three two one

**Constraints**

1 <= string length <= 200

**Sample Input 1**

i love india

**Sample Output 1**

india love i

**Solution:**

#include<stdio.h>

char \* strReverse(char \* str)

{

int start , end ;

char temp;

for(end = 0 ; str[end] ; end++);

for(start = 0 , --end ; start < end ; start++ , end--)

{

temp = str[start];

str[start] = str[end];

str[end] = temp;

}

return str;

}

char \* wordReversal(char \* str , char\* space)

{

int ind;

if(space[0] == 0)

return strReverse(str);

for(ind = (space - str)+1 ; str[ind] && str[ind] != 32 ; ind++);

wordReversal(space+1 , str+ind);

\*space = 0;

strReverse(str);

\*space = 32;

return str;

}

int main()

{

char str[200];

int ind;

scanf("%[^\n]s" , str);

strReverse(str);

for(ind = 0 ; str[ind] && str[ind] != 32 ; ind++);

wordReversal(str , str+ind);

printf("%s" , str);

return 0;

}

**Testcase 1:**

**Input:**

i love india

**Output:**

india love i

**Testcase 2:**

**Input:**

one two three

**Output:**

three two one

**Testcase 3:**

**Input:**

sdjgfvjdsv dfvgjdfvj jgv khjkfde yusfd ijkhfv

**Output:**

ijkhfv yusfd khjkfde jgv dfvgjdfvj sdjgfvjdsv

**Testcase 4:**

**Input:**

snbdcvdnsc dvcgdvc dhcvhgdvc hdgvchjvc hdgchjdcv hdcvhvc hsdvch hdc jdcbghj hjdchvhj hdbc hjdchj haai

**Output:**

haai hjdchj hdbc hjdchvhj jdcbghj hdc hsdvch hdcvhvc hdgchjdcv hdgvchjvc dhcvhgdvc dvcgdvc snbdcvdnsc

**Testcase 5:**

**Input:**

one two three four five six

**Output:**

six five four three two one

**Testcase 6:**

**Input:**

india delhi mumbai sivakasi virudhunagar cbe chennai ooty

**Output:**

ooty chennai cbe virudhunagar sivakasi mumbai delhi india

**Testcase 7:**

**Input:**

dxhcbndcv nd hjcv ysv hsvhjvssvsvsv cjhvcsv cjvjvcjd vcjvdjcv jvcjvdcvdvchjd vcjvdchjv djcvhjvchjvd adxiuahxkbsc xcjvwadfuwdsyu

**Output:**

xcjvwadfuwdsyu adxiuahxkbsc djcvhjvchjvd vcjvdchjv jvcjvdcvdvchjd vcjvdjcv cjvjvcjd cjhvcsv hsvhjvssvsvsv ysv hjcv nd dxhcbndcv

**Testcase 8:**

**Input:**

zxnb djbc djcvhjdvc hjdgc hdcg jdcg djcg uidgci dgci diu uidcgiugdc chgibcdc d

**Output:**

d chgibcdc uidcgiugdc diu dgci uidgci djcg jdcg hdcg hjdgc djcvhjdvc djbc zxnb

**Testcase 9:**

**Input:**

sdhgcjh jhcv shjcvhjcv

**Output:**

shjcvhjcv jhcv sdhgcjh

**Testcase 10:**

**Input:**

sjchb cbh dchjb

**Output:**

dchjb cbh sjchb

**Program 28**

**Sub Topic:**Number Crunching

**BINARY - DECIMAL**

 A positive integer is called Binary-Decimal it contains only 0’s and 1’s

**Sample Input0:**

32

**Sample Output0:**

11 11 10

3

**Explanation:**

There are many possibilities for representing 32 as a sum of Binary-Decimals

Few possibilities will be

10 + 10 + 1+ 1

Count = 5

11 + 10 + 10 + 1

Count = 4

11+11+10

Count = 3

The Expected output is(11 + 11 + 10) as it has minimum number of Binary-Decimals

(Count – 3)

**Sample Input 1**

32

**Sample Output 1**

11 11 10

3

**Solution:**

#include<stdio.h>

int nod(int num)

{

int spare = 0 ,digit , power , newnum = 0 ;

power = 1;

while(num/power)

{

digit= (num / power)%10;

spare = spare \* 10 + 1;

if(digit == 0 || digit == 9)

newnum = 0 \* power + newnum;

else

newnum = 1 \* power + newnum;

power \*= 10;

}

return (newnum == 0||spare<=num) ? spare : newnum;

}

int main()

{

int num , val , ctr , count = 0 ;;

scanf("%d" , &num);

do

{

val = nod(num);

if( val > num && num >= 10)

{

printf("%d" , num );

count++;

break;

}

if( num>=0 && num <= 9)

{

for(ctr = 1 ; ctr <= num ; ctr++)

{

printf("%d " , 1);

count++;

}

break;

}

while( (num-val)>=0 && num)

{

num -= val;

printf("%d " , val);

count++;

}

}while( num != 0 );

printf("\n%d" , count);

return 0;

}

**Testcase 1:**

**Input:**

32

**Output:**

11 11 10

3

**Testcase 2:**

**Input:**

434

**Output:**

111 111 111 101

4

**Testcase 3:**

**Input:**

10199

**Output:**

10100 11 11 11 11 11 11 11 11 11

10

**Testcase 4:**

**Input:**

42442

**Output:**

11111 11111 11111 1111 1111 1111 1111 1111 1111 1111 1111 111 110

13

**Testcase 5:**

**Input:**

4674

**Output:**

1111 1111 1111 1111 111 111 1 1 1 1 1 1 1 1

14

**Testcase 6:**

**Input:**

3654

**Output:**

1111 1111 1111 111 111 11 11 11 11 11 11 11 11 11

14

**Testcase 7:**

**Input:**

46754

**Output:**

11111 11111 11111 11111 1111 1111 11 11 11 11 11 11 11 11

14

**Testcase 8:**

**Input:**

16939

**Output:**

11111 1111 1111 1111 1111 1111 111 111 11 11 11 11 1 1 1 1 1 1 1

19

**Testcase 9:**

**Input:**

9990

**Output:**

1111 1111 1111 1111 1111 1111 1111 1111 1101 1

10

**Program 29**

**Sub Topic:**Strings

**DECOMPRESS THE STRING**

Assume that the given string has enough memory

Don't use any extra space(IN-PLACE)

**Sample Input 1**

a2b4c6

**Sample Output 1**

aabbbbcccccc

**Solution:**

#include<stdio.h>

int main()

{

char str[200] , ch;

int ind , count;

scanf("%s" , str);

int len , start;

for(len = 0 ; str[len] ; len++);

start = len ;

ind = count = 0;

while( ind < len )

{

ch = str[ind++];

while(str[ind] >= '0' && str[ind] <= '9' )

count = count \* 10 + (str[ind++] - '0');

while(count)

{

str[start++] = ch;

count--;

}

}

for(ind = len ; ind < start ; ind++)

str[ind - len] =str[ind];

str[ind - len] = 0;

printf("%s" , str);

return 0;

}

**Testcase 1:**

**Input:**

a2b4c6

Output:

Aabbbbcccccc

**Testcase 2:**

**Input:**

a12b3d4

Output:

Aaaaaaaaaaaabbbdddd

**Testcase 3:**

**Input:**

a100b3c12

**Output:**

Aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaabbbcccccccccccc

**Testcase 4:**

**Input:**

a76b23c10

**Output:**

Aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaabbbbbbbbbbbbbbbbbbbbbbbcccccccccc

**Testcase 5:**

**Input:**

x7y3d8

**Output:**

Xxxxxxxyyydddddddd

**Testcase 6:**

**Input:**

l123a3

**Output:**

Lllllllllllllllllllllllllllllllllllllllllllllllllllllllllllllllllllllllllllllllllllllllllllllllllllllllllllllllllllllllllllaaa

**Testcase 7:**

**Input:**

t6h7j9k3h6

**Output:**

Tttttthhhhhhhjjjjjjjjjkkkhhhhhh

**Testcase 8:**

**Input:**

j7b7m8v7b8

**Output:**

Jjjjjjjbbbbbbbmmmmmmmmvvvvvvvbbbbbbbb

**Testcase 9:**

**Input:**

j8b8n9j7m7v87

**Output:**

Jjjjjjjjbbbbbbbbnnnnnnnnnjjjjjjjmmmmmmmvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvv

**Program 30:**

**Sub Topic:** Branching and Looping

**Excel Sheet**

Given a number, convert it into corresponding alphabet.

**Input Output**

    1  A

   26  Z

   27  AA

   676  YZ

**Input Format**

Input is an integer

**Output Format**

Print the alphabets

**Constraints**

1 <= num <= 4294967295

**Sample Input 1**

26

**Sample Output 1**

Z

**Solution:**

#include <stdio.h>

#include<string.h>

int main() {

//code

unsigned int testnum=101027545,rem,ind,cases,tc,start,end;

char str[100],temp;

// scanf("%u",&cases);

// for(tc =0 ; tc < cases ; tc++)

// {

scanf("%u",&testnum);

ind=0;

while(testnum)

{

rem = (testnum% 26);

if(rem==0) {rem=26; testnum--;}

//printf("%c",rem+64);

str[ind++]=rem+64;

testnum/=26;

}

str[ind]='\0';

start=0;

end=ind-1;

while(start<end)

{

temp = str[start];

str[start]=str[end];

str[end]=temp;

start++;

end--;

}

//strrev(str);

printf("%s",str);

// }

return 0;

}

**Testcase 1:**

**Input:**

26

**Output:**

Z

**Testcase 2:**

**Input:**

27

**Output:**

AA

**Testcase 3:**

**Input:**

987654321

**Output:**

CECGIBQ

**Testcase 4:**

**Input:**

556

**Output:**

UJ

**Testcase 5:**

**Input:**

123456789

**Output:**

JJDDJA

**Testcase 6:**

**Input:**

16031994

**Output:**

AIBCYD

**Testcase 7:**

**Input:**

24031995

**Output:**

AZOHGM

**Testcase 8:**

**Input:**

6101965

**Output:**

MIDNY

**Testcase 9:**

**Input:**

1000

**Output:**

ALL

**Testcase 10:**

**Input:**

676

**Output:**

YZ

**Testcase 11:**

**Input:**

16384

**Output:**

XFD

**Testcase 12:**

**Input:**

27122005

**Output:**

BGICHA

**Program 31:**

**Sub Topic:** Arrays

**Frequency Sorting**

Given an array of integers arrange them in the descending order of their frequencies

**Sample Input 1**

11

1 2 3 1 1 1 3 2 4 4 2

**Sample Output 1**

1 1 1 1 2 2 2 3 3 4 4

**Solution:**

#include <stdio.h>

#include<limits.h>

#include<malloc.h>

#define SIZE sizeof(arr) / sizeof(arr[0])

int main()

{

int n , ind;

scanf("%d" , &n);

int arr[n] ;

int\*\* occurence = NULL;

int maxpos, count = 0, count\_flag, o\_row, index,max;

int newindex = 0, num, ctr;

for(ind = 0 ; ind < n ; ind++)

scanf("%d" , &arr[ind]);

occurence = (int\*\*)calloc(1,sizeof(int\*));

occurence[0] = (int\*)calloc(2,sizeof(int));

occurence[0][0] = arr[0];

occurence[0][1]++;

count++;

for(index = 1 ; index < SIZE ; index++)

{

//search in occurence array

for(o\_row = 0, count\_flag = 0 ; o\_row < count ; o\_row++)

{

if(occurence[o\_row][0] == arr[index])

{

occurence[o\_row][1]++;

count\_flag = 1;

}

} //search completed

if(count\_flag == 0)

{

occurence = (int\*\*)realloc(occurence,(count+1)\*sizeof(int\*));

occurence[count] = (int\*)calloc(2,sizeof(int));

occurence[count][0] = arr[index];

occurence[count][1]++;

count++;

}

}

for(index = 0 ; index < count ; index++)

{

//find maximum in occurence

for(o\_row = 0, max = INT\_MIN ; o\_row < count ; o\_row++)

{

if(occurence[o\_row][1] > max)

{

max = occurence[o\_row][1];

maxpos = o\_row;

}

}

num = occurence[maxpos][0];

for(ctr = 1 ; ctr <= max ; ctr++)

arr[newindex++] = num;

occurence[maxpos][1] = -1;

}

//printf("%d %d\n",max, maxpos);

// for(index = 0 ; index < count ; index++)

//printf("%d ,%d\n", occurence[index][0], occurence[index][1]);

for(index = 0 ; index < SIZE ; index++)

printf("%d ",arr[index]);

for(index=0 ; index < count ; index++)

free(occurence[index]);

free(occurence);

return 0;

}

**Testcase 1:**

**Input:**

11

1 2 3 1 1 1 3 2 4 4 2

**Output;**

1 1 1 1 2 2 2 3 3 4 4

**Testcase 2:**

**Input:**

6

-1 -1 4 5 2 2

**Output:**

-1 -1 2 2 4 5

**Testcase 3:**

**Input:**

12

4 4 2 1 3 4 1 3 4 5 6 1

**Output:**

4 4 4 4 1 1 1 3 3 2 5 6

**Testcase 4:**

**Input:**

15

1 2 3 4 5 6 2 3 5 2 5 6 3 2 3

**Output:**

2 2 2 2 3 3 3 3 5 5 5 6 6 1 4

**Testcase 5:**

**Input:**

5

1 2 3 4 5

**Output:**

1 2 3 4 5

**Testcase 6:**

**Input:**

12

1 2 3 2 23 123 33 23 87 23 83 9

**Output:**

23 23 23 2 2 1 3 123 33 87 83 9

**Testcase 7:**

**Input:**

5

12345 12345 24 43 3

**Output:**

12345 12345 24 43 3

**Testcase 8:**

**Input:**

10

10 39 349 27 9 10 9 7 9 10

**Output;**

10 10 10 9 9 9 39 349 27 7

**Testcase 9:**

**Input:**

5

3 3 2 5 3

**Output:**

3 3 3 2 5

**Program 32:**

**Sub Topic:**Arrays

**LATIN SQUARE**

write a program to construct a latin square of a given N without using

i) any conditional statements ( if - else / ternary operator)

ii) matrix

A latin square is an n X n matrix array filled with n different symbols , each occurring exactly once in each row and exactly once in each column

**Sample Input 1**

3

**Sample Output 1**

A B C

C A B

B C A

**Solution:**

#include<stdio.h>

#include<malloc.h>

void setString(char \*str , int N)

{

int ctr;

for(ctr = 0 ; ctr < N ; ctr++)

str[ctr] = ctr +65;

str[ctr] = 0;

}

void strRev(char \*str)

{

int start , end;

char temp;

for(end = 0 ; str[end] ; end++);

for(start = 0 , --end ; start < end ; start++ , end--)

{

temp = str[start];

str[start] = str[end];

str[end] = temp;

}

}

void strRotate(char \*str)

{

strRev(str);

strRev(str + 1);

}

void strCopy(char \*s1 , char \*s2)

{

int ind;

for(ind = 0 ; s2[ind] ; s1[ind] = s2[ind] , ind++);

s1[ind] = 0 ;

}

int main()

{

int N , row , col;

scanf("%d" ,&N);

char arr[N][N+1];

char str[N+1];

setString(str , N);

for(row = 0 ; row < N ; row++ , printf("\n"))

{

strCopy(arr[row] , str) ;

for(col = 0 ; col < N ; col++)

printf("%c " , arr[row][col]);

printf("\n");

strRotate(str);

}

return 0;

}

**Testcase 1:**

**Input:**

3

**Output:**

A B C

C A B

B C A

**Testcase 2:**

**Input:**

4

**Output:**

A B C D

D A B C

C D A B

B C D A

**Testcase 3:**

**Input:**

4

**Output:**

A B C D

D A B C

C D A B

B C D A

**Testcase 4:**

**Input:**

5

**Output:**

A B C D E

E A B C D

D E A B C

C D E A B

B C D E A

**Testcase 5:**

**Input:**

2

**Output:**

A B

B A

**Testcase 6:**

**Input:**

7

**Output:**

A B C D E F G

G A B C D E F

F G A B C D E

E F G A B C D

D E F G A B C

C D E F G A B

B C D E F G A

**Testcase 7:**

**Input:**

9

**Output:**

A B C D E F G H I

I A B C D E F G H

H I A B C D E F G

G H I A B C D E F

F G H I A B C D E

E F G H I A B C D

D E F G H I A B C

C D E F G H I A B

B C D E F G H I A

**Testcase 8:**

**Input:**

11

**Output:**

A B C D E F G H I J K

K A B C D E F G H I J

J K A B C D E F G H I

I J K A B C D E F G H

H I J K A B C D E F G

G H I J K A B C D E F

F G H I J K A B C D E

E F G H I J K A B C D

D E F G H I J K A B C

C D E F G H I J K A B

B C D E F G H I J K A

**Testcase 9:**

**Input:**

6

**Output:**

A B C D E F

F A B C D E

E F A B C D

D E F A B C

C D E F A B

B C D E F A

**Program 33:**

**Sub Topic:** Algorithm

**PATTERN**

**Sample Input 1**

5

**Sample Output 1**

1

1 1

2 1

1 2 1 1

1 1 1 2 2 1

**Solution:**

#include<stdio.h>

#include<malloc.h>

int main()

{

int N , row , col , ctr ,count , val , ind;

scanf("%d" ,&N);

int \*\*arr;

arr = (int\*\*) calloc(N , sizeof(int\*));

for(ctr = 0 ; ctr < N ; ctr++)

arr[ctr] = (int\*)calloc(N\*N/4 , sizeof(int));

arr[0][0] = 1;

ind = 0;

val = 1;

count = 0;

for(row = 0 ; row < N-1 ; row++ , ind = 0 )

{

for(col = 0 ; arr[row][col] && col < (N\*N/4)-1 ; col++)

{

if(arr[row][col] == arr[row][col+1])

count++;

else if(arr[row][col+1] == 0 && count == 0) // last mismatch

{

arr[row+1][ind] = 1;

arr[row+1][ind+1] = arr[row][col];

count = 0;

ind += 2;

}

else

{

arr[row+1][ind] = count + 1;

arr[row+1][ind+1] = arr[row][col];

count = 0;

ind += 2;

}

}

}

for(row = 0 ; row < N ; row++ , printf("\n"))

{

for(col = 0 ; arr[row][col] && col < N\*N/4 ; col++)

printf("%d " , arr[row][col]);

}

return 0;

}

**Testcase 1:**

**Input:**

5

**Output:**

1

1 1

2 1

1 2 1 1

1 1 1 2 2 1

**Testcase 2:**

**Input:**

3

**Output:**

1

1 1

**Testcase 3:**

**Input:**

6

**Output:**

1

1 1

2 1

1 2 1 1

1 1 1 2 2 1

3 1 2 2 1 1

**Testcase 4:**

**Input:**

4

**Output;**

1

1 1

2 1

1 2 1 1

**Testcase 5:**

**Input:**

7

**Output:**

1

1 1

2 1

1 2 1 1

1 1 1 2 2 1

3 1 2 2 1 1

1 3 1 1 2 2 2 1

**Testcase 6:**

**Input:**

8

**Output:**

1

1 1

2 1

1 2 1 1

1 1 1 2 2 1

3 1 2 2 1 1

1 3 1 1 2 2 2 1

1 1 1 3 2 1 3 2 1 1

**Testcase 7:**

**Input:**

9

**Output:**

1

1 1

2 1

1 2 1 1

1 1 1 2 2 1

3 1 2 2 1 1

1 3 1 1 2 2 2 1

1 1 1 3 2 1 3 2 1 1

3 1 1 3 1 2 1 1 1 3 1 2 2 1

**Testcase 8:**

**Input:**

10

**Output:**

1

1 1

2 1

1 2 1 1

1 1 1 2 2 1

3 1 2 2 1 1

1 3 1 1 2 2 2 1

1 1 1 3 2 1 3 2 1 1

3 1 1 3 1 2 1 1 1 3 1 2 2 1

1 3 2 1 1 3 1 1 1 2 3 1 1 3 1 1 2 2 1 1

**Testcase 9:**

**Input:**

11

**Output:**

1

1 1

2 1

1 2 1 1

1 1 1 2 2 1

3 1 2 2 1 1

1 3 1 1 2 2 2 1

1 1 1 3 2 1 3 2 1 1

3 1 1 3 1 2 1 1 1 3 1 2 2 1

1 3 2 1 1 3 1 1 1 2 3 1 1 3 1 1 2 2 1 1

1 1 1 3 1 2 2 1 1 3 3 1 1 2 1 3 2 1 1 3 2 1 2 2 2 1

**Program 34**

**Sub Topic:** Strings

**Roman to Decimal**

Given a Roman numeral, find its corresponding decimal value.

**Input Format**

Input is a string which contains Roman numbers

**Output Format**

Print the decimal value

**Constraints**

1<=string\_length<100

**Sample Input 1**

XLV

**Sample Output 1**

45

**Solution:**

#include<stdio.h>

//#include<conio.h>

int getValue( char ch)

{

if( ch == 'I' )

return 1;

if( ch == 'V' )

return 5;

if( ch == 'X' )

return 10;

if( ch == 'L' )

return 50;

if( ch == 'C' )

return 100;

if( ch == 'D' )

return 500;

if( ch == 'M' )

return 1000;

}

int main()

{

char str[100]="";

int num=0,ctr,v1,v2,count=0;

scanf("%s",str);

for( ctr = 0 ; str[ctr] != '\0'; )

{

v1 = getValue(str[ctr]);

if( str[ctr+1] != '\0')

{

v2 = getValue(str[ctr+1]);

if( v1 < v2)

{

num += (v2-v1);

ctr+=2;

}else

{

num += v1;

ctr++;

}

}

else {

num += v1; ctr++;

}

}

printf("%d",num);

}

**Testcase 1:**

**Input:**

XLV

**Output:**

45

**Testcase 2:**

**Input:**

XVII

**Output:**

17

**Testcase 3:**

**Input:**

MMMDCCCXCVIII

**Output:**

3898

**Testcase 4:**

**Input:**

MMMCMXXXIX

**Output:**

3939

**Testcase 5:**

**Input:**

MMMDCCCLXXXVIII

**Output:**

3888

**Testcase 6:**

**Input:**

MMMMCDXCVIII

**Output:**

4498

**Testcase 7:**

**Input:**

MMCXCV

**Output:**

2195

**Testcase 8:**

**Input:**

MCCCXCVIII

**Output:**

1398

**Testcase 9:**

**Input:**

CMXLVIII

**Output:**

948

**Testcase 10:**

**Input:**

CMXV

**Output:**

915

**Testcase 11:**

**Input:**

MMMMDCCCXXVIII

**Output:**

4828

**Testcase 12:**

**Input:**

MMMDCXLIX

**Output:**

3649

**Program 35:**

**Sub Topic:** Algorithm

**SNAKE PATTERN**

( DONT use MATRIX)

**Input Format**

N = 5

**Output Format**

**Sample Input 1**

5

**Sample Output 1**

1 2 3 4 5

10 9 8 7 6

11 12 13 14 15

20 19 18 17 16

21 22 23 24 25

**Solution:**

#include<stdio.h>

int main()

{

int row , col , N , count , val, spacecount;

scanf("%d" , &N);

for(row = 1, count = 0 , spacecount = (N-1); row <= N ; printf("\n") , row++,spacecount--)

{

for(col = 1 ; col <= spacecount ;printf(" "),col++);

if(row % 2 == 0)

{

val = count \* N ;

for(col = 1 ; col <= N ; printf("%\*d " , 2, val--),col++);

}

else

{

val = count \* N + 1 ;

count+=2;

for(col = 1 ; col <= N ; printf("%\*d " , 2,val++),col++);

}

}

return 0;

}

**Testcase 1:**

**Input:**

5

**Output:**

1 2 3 4 5

10 9 8 7 6

11 12 13 14 15

20 19 18 17 16

21 22 23 24 25

**Testcase 2:**

**Input:**

6

**Output:**

1 2 3 4 5 6

12 11 10 9 8 7

13 14 15 16 17 18

24 23 22 21 20 19

25 26 27 28 29 30

36 35 34 33 32 31

**Testcase 3:**

**Input:**

3

**Output:**

1 2 3

6 5 4

7 8 9

**Testcase 4:**

**Input:**

7

**Output:**

1 2 3 4 5 6 7

14 13 12 11 10 9 8

15 16 17 18 19 20 21

28 27 26 25 24 23 22

29 30 31 32 33 34 35

42 41 40 39 38 37 36

43 44 45 46 47 48 49

**Testcase 5:**

**Input:**

4

**Output:**

1 2 3 4

8 7 6 5

9 10 11 12

16 15 14 13

**Testcase 6:**

**Input:**

9

**Output:**

1 2 3 4 5 6 7 8 9

18 17 16 15 14 13 12 11 10

19 20 21 22 23 24 25 26 27

36 35 34 33 32 31 30 29 28

37 38 39 40 41 42 43 44 45

54 53 52 51 50 49 48 47 46

55 56 57 58 59 60 61 62 63

72 71 70 69 68 67 66 65 64

73 74 75 76 77 78 79 80 81

**Program 36:**

**Sub Topic:**Strings

**SPLIT THE STRING ACCORDING TO DICTIONARY**

Given an input string and a dictionary of word,find out if the input string can be segmented into a space separated sequence of dictionary words

Consider the following dictionary

{i , like , ice , cream , icecream }

the input string is ilikeicecream the expected output is

i like icecream

**Input Format**

N - no of words in dictionary

dictionary of words

input string

**Output Format**

display the separated input string

**Sample Input 1**

5

i like ice cream icecream

ilikeicecream

**Sample Output 1**

i like icecream

**Solution:**

#include<stdio.h>

int StrLen(char \*str)

{

int ind;

for(ind = 0 ; str[ind] ; ind++);

return ind;

}

int strCmp(char \*s1 , char \*s2)

{

int ind;

for(ind = 0 ; s1[ind] && s2[ind] && s1[ind] == s2[ind] ; ind++);

return s1[ind] - s2[ind];

}

int main()

{

int N , ind , ind1 , len , safeind1, prevlen;

char safe , \*start ;

scanf("%d" , &N);

char str[N][50];

char input[50];

for(ind = 0 ; ind < N ; ind++)

scanf("%s" , str[ind]);

scanf("%s" , input);

start = input;

prevlen = -1;

for( ind =0 ; input[ind] ; ind++ , prevlen = - 1)

{

for(ind1 = 0 ; ind1 < N ; ind1++)

{

len = StrLen(str[ind1]);

safe = input[ind+len];

input[ind+len] = 0;

if( strCmp(input+ind , str[ind1]) == 0 && prevlen < len)

{

safeind1 = ind1;

prevlen = len;

}

input[ind+len] = safe;

}

printf("%s " , str[safeind1]);

ind = ind + StrLen(str[safeind1]) - 1;

}

return 0 ;

}

**Testcase 1:**

**Input:**

5

i like ice cream icecream

ilikeicecream

**Output:**

i like icecream

**Testcase 2:**

**Input:**

8

hai hello how are you i am good

haihellohowareyouiamgood

**Output:**

hai hello how are you i am good

**Testcase 3:**

**Input:**

5

i love sum sung sumsung

ilovesumsung

**Output:**

i love sumsung

**Testcase 4:**

**Input:**

5

love india delhi in i

iloveindiaindelhi

**Output:**

i love india in delhi

**Testcase 5:**

**Input:**

7

i mobile phone love apple android i7

iloveapplei7phone

**Output:**

i love apple i7 phone

**Testcase 6:**

**Input:**

6

lovely love loves loved being bee

lovelyloveloveslovedbeing

**Output;**

lovely love loves loved being

**Testcase 7:**

**Input:**

5

wonder wonderful rain bow rainbow

wonderfulrainbow

**Output:**

wonderful rainbow

**Testcase 8:**

**Input:**

3

hai haai haaai

haaaihaai

**Output:**

haaai haai

**Testcase 9:**

**Input:**

3

lovely dear loving

lovelydear

Output:

lovely dear