ICT 1018 Assignment – Documentation

Lenise Silvio

Question 1

Time complexity:

	Best Case	Average Case	Worst Case
Shell Sort	O(n)	$O(n^2)$	$O(n^2)$
Quick Sort	$O(n\log_2 n)$	$O(n\log_2 n)$	$O(n^2)$

The function `shell_sort` performs the shell sort algorithm. The algorithm for the quick sort algorithm is performed using the `quick_sort` and `partition` functions. The function `swap` is used to help swap elements in quick sort. The `main` contains the test driver that tests the execution of both algorithms. The function `random_array` populates the arrays with random numbers. The function `print_array` is used to display the contents of the array for testing purposes.

Shell sort adapted from: https://www.geeksforgeeks.org/shellsort/
Quick sort adapted from: https://www.geeksforgeeks.org/quicksort-using-random-pivoting/

Program:

```
import java.util.Random;

public class Question1 {
    public static void main(String[] args) {
        /* Test Driver */
        int[] A = new int[256]; // first array
        random_array(256, A); // populate array with random numbers
        int[] B = new int[257]; // create second array
        random_array(257, B); // populate array with random numbers

        // Perform shell sort on array A
        System.out.println("Array A before shell sort:");
        print_array(A, 256);
        shell_sort(A, 256);
        System.out.println();
        System.out.println("Array A after shell sort: ");
        print_array(A, 256);

        System.out.println();
        System.out.println();

        // Perform quick sort on array B
        System.out.println("Array B before quick sort:");
        print_array(B, 257);
        quick_sort(B, 0, 256);
        System.out.println();
        System.out.println("Array B after quick sort: ");
        print_array(B, 257);

    }
```

```
    //prints array
    static void print_array(int arr[], int size) {
        for (int i = 0; i < size; i++) {
            System.out.print(arr[i] + " ");
        }
        System.out.println();
    }

    // Populates an array with random numbers from 0 to 1024
    static void random_array(int n, int arr[]) {
        Random rn = new Random();

        // generate random number for each index in the array
        for (int i = 0; i < n; i++) {
            arr[i] = rn.nextInt(0, 1024);
        }
    }

    //Shell sort algorithm
    // Adapted from: https://www.geeksforgeeks.org/shellsort/
    static void shell_sort(int A[], int size) {
        int gap = size / 2; // sets initial gap size to half the array
length

        // while there is a gap of at least 1 between each element
        while (gap > 0) {
            // perform insertion sort but with the specified gap
            for (int i = gap;  i < size; i++) {
                int j = i;
                int temp = A[i];
                // find correct position for A[i]
                // compare elements at the specified gap
                // shift elements larger than temp down the array by the
specified gap
                while (j >= gap) {
                    if (temp > A[j-gap]) { // break if temp is in its
correct position
                        break;
                    }
                    A[j] = A[j-gap];
                    j -= gap;
                }
                A[j] = temp; // put element a[i] to its correct position
            }
            gap = gap / 2; // half the gap after each iteration
        }
    }

    // Quick sort algorithm
    // Adapted from: https://www.geeksforgeeks.org/quicksort-using-random-
pivoting/
    static void quick_sort(int array[], int start, int end) {
        if (start < end) { //continuously sort array while partitioned
array is larger than one
            Random rn = new Random();
            swap(array, rn.nextInt(end - start) + start, end); // choose
random number as pivot and put it lass
            int p = partition(array, start, end); // call partition
function
            quick_sort(array, start, p - 1); //recursively call quick_sort
function on left part of previously partitioned array
```

```
            quick_sort(array, p + 1, end); //recursively call quick_sort
function on right part of previously partitioned array
        }
    }

    // Partition function for quick sort
    static int partition(int array[], int start, int end) {
        int pivot = array[end]; // find the pivot

        int x = start; //set to the starting index of the partition

        for (int i = start; i < end; i++) {
            if (array[i] <= pivot) { // put elements smaller than or equal
to the pivot to the left
                swap(array, i, x);
                x++;
            }
        }
        swap(array, x, end); //put pivot to its correct place
        return x;
    }


    // swaps to elements in an array
    static void swap(int array[], int index1, int index2) {
        int temp = array[index1]; // save index temporarily for swap
        array[index1] = array[index2]; //set element at index1 equal to
element at index2
        array[index2] = temp; //set element at index 2 as being equal to
the temp variable
    }
}
```

Outputs:

```
/Library/Java/JavaVirtualMachines/temurin-17.jdk/Contents/Home/bin/java -javaagent:/Applications/IntelliJ IDEA.app/Contents/lib/idea_rt.jar=56723:/Applications/IntelliJ
 IDEA.app/Contents/bin -Dfile.encoding=UTF-8 -classpath /Users/lenisesilvio/IdeaProjects/ICT1018_Assignment/out/production/Question1 Question1
Array A before shell sort:
747 767 118 338 108 900 946 237 216 698 931 212 597 1017 287 336 616 237 710 36 478 425 393 189 750 513 368 935 415 24 663 323 615 55 633 951 788 480 368 37 495 546 124
 798 798 815 835 89 74 299 600 415 641 70 447 265 604 12 944 622 612 1019 923 22 945 341 364 690 212 568 271 144 758 491 566 10 320 634 89 319 637 900 766 773 254 594 570
  797 190 812 887 424 965 999 630 986 208 962 663 493 49 115 88 87 756 123 795 62 67 706 811 811 267 863 0 510 478 438 706 801 795 967 267 0 843 709 1009 440 974 230 835
 270 950 121 273 418 242 60 369 99 942 239 640 325 1018 36 167 49 425 89 799 317 635 584 182 29 638 892 210 728 688 551 182 958 489 562 260 664 1015 918 898 289 778 490
 288 693 850 316 57 840 230 35 793 409 999 788 703 166 741 244 483 98 896 239 57 420 338 391 503 355 780 239 917 800 478 666 467 675 796 526 512 233 30 765 632 92 103 661
  877 68 520 534 1002 893 70 677 789 618 122 608 436 41 312 689 1009 775 947 999 865 54 864 514 322 701 984 717 260 431 258 306 778 415 369 457 945 338

Array A after shell sort:
0 0 10 12 22 24 29 30 35 36 36 37 41 49 49 54 55 57 57 60 62 67 68 70 70 74 87 88 89 89 89 92 98 99 103 108 115 118 121 122 123 124 144 166 167 182 182 189 190 208 210
 212 212 216 230 230 233 237 237 239 239 239 242 244 254 258 260 260 265 267 267 270 271 273 287 288 289 299 306 312 316 317 319 320 322 323 325 336 338 338 338 341 355
 364 368 368 369 369 391 393 409 415 415 415 418 420 424 425 425 431 436 438 440 447 457 467 478 478 478 480 483 489 490 491 493 495 503 510 512 513 514 520 526 534 546
 551 562 566 568 570 584 594 597 600 604 608 612 615 616 618 622 630 632 633 634 635 637 638 640 641 661 663 663 664 666 675 677 688 689 690 693 698 701 703 706 706 709
 710 717 728 741 747 750 756 758 765 766 767 773 775 778 778 780 788 788 789 793 795 795 796 797 798 798 799 800 801 811 811 812 815 835 835 840 843 850 863 864 865 877
 887 892 893 896 898 900 900 917 918 923 931 935 942 944 945 945 946 947 950 951 958 962 965 967 974 984 986 999 999 999 1002 1009 1009 1015 1017 1018 1019


Array B before quick sort:
896 80 309 223 264 901 44 60 382 200 846 321 329 604 915 481 852 1001 582 991 513 604 284 332 860 523 903 334 148 795 332 290 166 37 595 355 602 791 957 907 371 88 571
 954 801 47 413 694 442 613 425 51 937 40 46 339 422 49 88 879 39 297 836 791 21 584 470 855 707 575 970 316 283 88 9 918 537 986 573 766 125 433 423 538 956 78 303 543
 55 460 188 919 686 83 965 867 759 193 296 3 490 902 58 789 671 975 798 795 731 230 602 385 710 472 423 221 994 855 520 888 824 994 285 524 723 546 440 719 626 407 432
 176 888 371 245 735 39 453 484 147 525 730 733 263 957 736 273 964 450 459 160 294 720 891 482 792 829 30 4 545 943 723 215 676 789 655 484 611 986 324 707 580 88 319
 172 88 650 209 846 536 774 49 735 753 809 416 884 769 659 974 401 547 981 636 4 809 105 354 750 871 994 338 33 632 611 924 177 572 597 948 965 799 340 786 685 161 105
 857 569 529 894 699 328 923 14 667 159 901 795 115 1019 949 613 247 104 886 216 378 979 379 581 913 478 547 597 757 499 510 697 429 667 143 166 620 528 851 384

Array B after quick sort:
3 4 4 9 14 21 30 33 37 39 39 40 44 46 47 49 49 51 55 58 60 78 80 83 88 88 88 88 88 104 105 105 115 125 143 147 148 159 160 161 166 166 172 176 177 188 193 200 209 215 216
  221 223 230 245 247 263 264 273 283 284 285 290 294 296 297 303 309 316 319 321 324 328 329 332 332 334 338 339 340 354 355 371 371 378 379 382 384 385 401 407 413 416
 422 423 423 425 429 432 433 440 442 450 453 459 460 470 472 478 481 482 484 484 490 499 510 513 520 523 524 525 528 529 536 537 538 543 545 546 547 547 569 571 572 573
 575 580 581 582 584 595 597 597 602 602 604 604 611 611 613 613 620 626 632 636 650 655 659 667 667 671 676 685 686 694 697 699 707 707 710 719 720 723 723 730 731 733
 735 735 736 750 753 757 759 766 769 774 786 789 789 791 791 792 795 795 795 798 799 801 809 809 824 829 836 846 846 851 852 855 855 857 860 867 871 879 884 886 888 888
 891 894 896 901 901 902 903 907 913 915 918 919 923 924 937 943 948 949 954 956 957 957 964 965 965 970 974 975 979 981 986 986 991 994 994 994 1001 1019

Process finished with exit code 0
```

```
/Library/Java/JavaVirtualMachines/temurin-17.jdk/Contents/Home/bin/java -javaagent:/Applications/IntelliJ IDEA.app/Contents/lib/idea_rt.jar=56725:/Applications/IntelliJ IDEA.app/Contents/bin -Dfile.encoding=UTF-8 -classpath /Users/lenisesilvio/IdeaProjects/ICT1018_Assignment/out/production/Question1 Question1
Array A before shell sort:
572 134 414 477 466 654 271 933 121 290 704 203 253 594 992 595 628 951 251 885 437 215 655 161 522 973 735 707 292 41 262 711 591 21 334 351 230 952 706 771 520 734 513
 57 395 216 341 534 560 672 515 772 280 833 740 964 119 629 52 182 390 399 169 856 446 667 967 12 834 451 370 29 156 243 708 3 940 448 32 486 988 389 135 96 920 975 553
 391 253 762 63 554 464 795 535 345 689 800 880 42 399 54 519 442 862 874 113 882 263 402 555 395 928 366 239 979 988 9 840 688 197 869 104 101 424 854 313 660 847 174
 689 3 117 666 362 674 374 587 625 460 838 216 349 205 419 866 931 420 142 376 297 9 246 445 536 176 182 200 502 207 450 587 1011 478 156 118 130 413 523 413 43 603 15 23
  927 884 527 105 268 538 84 420 690 600 698 11 694 84 676 504 54 468 48 767 413 2 683 339 55 371 71 360 263 101 267 986 728 328 266 397 401 383 471 532 717 615 823 86
 939 151 792 366 993 873 194 467 535 185 844 184 370 236 848 437 110 820 993 167 240 529 701 979 411 26 959 81 98 253 918 167 605 707 443 435 224 856

Array A after shell sort:
2 3 3 9 9 11 12 15 21 23 26 29 32 41 42 43 48 52 54 54 55 57 63 71 81 84 84 86 96 98 101 101 104 105 110 113 117 118 119 121 130 134 135 142 151 156 156 161 167 167 169
 174 176 182 182 184 185 194 197 200 203 205 207 215 216 216 224 230 236 239 240 243 246 251 253 253 253 262 263 263 266 267 268 271 280 290 292 297 313 328 334 339 341
 345 349 351 360 362 366 366 370 370 371 374 376 383 389 390 391 395 395 397 399 399 401 402 411 413 413 413 414 419 420 420 424 435 437 437 442 443 445 446 448 450 451
 460 464 466 467 468 471 477 478 486 502 504 513 515 519 520 522 523 527 529 532 534 535 535 536 538 553 554 555 560 572 587 587 591 594 595 600 603 605 615 625 628 629
 654 655 660 666 667 672 674 676 683 688 689 689 690 694 698 701 704 706 707 707 708 711 717 728 734 735 740 762 767 771 772 792 795 800 820 823 833 834 838 840 844 847
 848 854 856 856 862 866 869 873 874 880 882 884 885 918 920 927 928 931 933 939 940 951 952 959 964 967 973 975 979 979 986 988 988 992 993 993 1011


Array B before quick sort:
332 431 845 321 197 678 123 324 659 880 321 819 305 8 58 792 328 700 682 838 45 616 158 942 686 998 850 624 14 708 466 437 333 311 995 699 171 848 802 560 367 365 622 452
  996 497 899 761 420 567 743 829 536 707 520 313 830 698 934 875 685 660 622 1010 726 527 487 424 139 358 401 629 197 450 587 738 673 241 501 748 428 85 341 26 95 798 62
  209 748 955 1006 231 571 141 952 885 815 989 994 496 556 784 720 122 733 280 97 885 50 46 920 601 168 701 638 967 716 929 71 734 513 400 435 460 896 472 951 423 928 586
  92 892 855 630 486 680 512 24 55 102 861 721 117 169 480 793 746 44 293 115 812 966 195 739 637 258 836 925 116 1006 369 254 476 563 466 447 475 870 116 871 342 668 748
  1 148 323 156 783 801 815 820 446 500 1011 891 602 979 370 591 771 1014 949 490 929 746 509 870 789 440 284 130 944 838 768 422 744 1020 912 36 72 938 698 192 936 424
 972 626 670 830 515 361 308 70 609 595 474 528 707 475 158 133 562 115 410 398 632 595 684 694 758 960 108 269 181 401 602 847 954 494 599 582 413 347 854 876 348 1012

Array B after quick sort:
1 8 14 24 26 36 44 45 46 50 55 58 62 70 71 72 85 92 95 97 102 108 115 115 116 116 117 122 123 130 133 139 141 148 156 158 158 168 169 171 181 192 195 197 197 209 231 241
 254 258 269 280 284 293 305 308 311 313 321 321 323 324 328 332 333 341 342 347 348 358 361 365 367 369 370 398 400 401 401 410 413 420 422 423 424 424 428 431 435 437
 440 446 447 450 452 460 466 466 472 474 475 475 476 480 486 487 490 494 496 497 500 501 509 512 513 515 520 527 528 536 556 560 562 563 567 571 582 586 587 591 595 595
 599 601 602 602 609 616 622 622 624 626 629 630 632 637 638 659 660 668 670 673 678 680 682 684 685 686 694 698 698 699 700 701 707 707 708 716 720 721 726 733 734 738
 739 743 744 746 746 748 748 748 758 761 768 771 783 784 789 792 793 798 801 802 812 815 815 819 820 829 830 830 836 838 838 845 847 848 850 854 855 861 870 870 871 875
 876 880 885 885 891 892 896 899 912 920 925 928 929 929 934 936 938 942 944 949 951 952 954 955 960 966 967 972 979 989 994 995 996 998 1006 1006 1010 1011 1012 1014
 1020

Process finished with exit code 0
```

```
/Library/Java/JavaVirtualMachines/temurin-17.jdk/Contents/Home/bin/java -javaagent:/Applications/IntelliJ IDEA.app/Contents/lib/idea_rt.jar=56792:/Applications/IntelliJ IDEA.app/Contents/bin -Dfile.encoding=UTF-8 -classpath /Users/lenisesilvio/IdeaProjects/ICT1018_Assignment/out/production/Question1 Question1
Array A before shell sort:
909 455 269 689 719 907 935 23 124 543 393 251 855 228 160 18 625 745 580 462 495 601 921 756 17 820 57 108 623 322 783 87 900 323 389 102 112 363 225 361 761 268 492 38
 978 917 724 876 288 698 851 871 313 1012 783 564 430 300 580 439 651 178 159 684 477 115 621 252 883 107 1012 337 233 88 801 1004 482 175 867 181 148 723 981 147 648 132
  930 1022 418 661 164 721 351 737 78 356 94 383 716 730 545 308 104 182 276 809 303 948 205 477 724 470 148 860 211 444 550 689 729 824 182 152 354 289 52 888 164 103
 208 655 427 104 705 110 452 129 577 168 31 742 988 198 431 182 234 125 853 841 90 290 572 716 944 93 646 528 80 541 39 926 252 459 529 14 922 705 913 408 826 1021 1016
 235 600 785 125 829 349 606 98 419 363 641 852 418 858 780 535 773 764 632 354 752 680 346 775 52 758 213 343 229 710 79 195 298 903 615 966 851 437 294 258 304 419 372
 937 672 122 460 542 360 385 51 731 581 926 869 916 835 796 850 668 847 942 200 648 779 691 112 17 737 442 254 574 602 329 164 194 176 986 455 162 673 293 756 795 390

Array A after shell sort:
14 17 17 18 23 31 38 39 51 52 52 57 78 79 80 87 88 90 93 94 98 102 103 104 104 107 108 110 112 112 115 122 124 125 125 129 132 147 148 148 152 159 160 162 164 164 164 168
  175 176 178 181 182 182 182 194 195 198 200 205 208 211 213 225 228 229 233 234 235 251 252 252 254 258 268 269 276 288 289 290 293 294 298 300 303 304 308 313 322 323
 329 337 343 346 349 351 354 354 356 360 361 363 363 372 383 385 389 390 393 408 418 418 419 419 427 430 431 437 439 442 444 452 455 455 459 460 462 470 477 477 482 492
 495 528 529 535 541 542 543 545 550 564 572 574 577 580 580 581 600 601 602 606 615 621 623 625 632 641 646 648 648 651 655 661 668 672 673 680 684 689 689 691 698 705
 705 710 716 716 719 721 723 724 724 729 730 731 737 737 742 745 752 756 756 758 761 764 773 775 779 780 783 783 785 795 796 801 809 820 824 826 829 835 841 847 850 851
 851 852 853 855 858 860 867 869 871 876 883 888 900 903 907 909 913 916 917 921 922 926 926 930 935 937 942 944 948 966 978 981 986 988 1004 1012 1012 1016 1021 1022


Array B before quick sort:
429 688 509 674 111 935 423 884 753 100 185 294 704 706 997 200 426 162 1003 108 861 694 466 28 334 156 885 526 220 558 64 533 948 536 571 91 630 426 840 845 408 494 680
 213 139 509 707 718 795 654 808 692 283 719 721 155 64 631 684 354 77 284 86 40 368 705 297 765 236 434 376 732 1015 404 24 729 228 951 547 662 645 175 760 1019 458 802
 133 37 429 994 968 927 338 829 402 463 105 916 909 946 470 298 385 459 148 829 722 344 726 602 76 199 878 1005 547 871 370 791 564 98 540 708 657 235 242 251 60 435 550
 888 561 928 831 375 169 960 504 722 157 635 359 78 904 520 353 304 881 34 28 535 808 762 534 261 719 367 54 702 156 636 48 341 476 1009 813 3 58 101 682 58 832 106 927
 1017 542 770 996 353 426 338 464 904 680 405 547 141 74 338 37 237 761 721 14 19 853 459 936 565 379 864 306 861 133 333 136 965 598 93 640 360 918 544 661 725 1019 474
 714 837 714 190 456 1008 925 347 841 667 870 598 697 373 435 89 26 35 563 843 807 537 632 705 188 555 1022 435 628 867 619 370 147 788 536 985 550 47 719 195 873

Array B after quick sort:
3 14 19 24 26 28 28 34 35 37 37 40 47 48 54 58 58 60 64 64 74 76 77 78 86 89 91 93 98 100 101 105 106 108 111 133 133 136 139 141 147 148 155 156 156 157 162 169 175 185
 188 190 195 199 200 213 220 228 235 236 237 242 251 261 283 284 294 297 298 304 306 333 334 338 338 338 341 344 347 353 353 354 359 360 367 368 370 370 373 375 376 379
 385 402 404 405 408 423 426 426 426 429 429 434 435 435 435 456 458 459 459 463 464 466 470 474 476 494 504 509 509 520 526 533 534 535 536 536 537 540 542 544 547 547
 547 550 550 555 558 561 563 564 565 571 598 598 602 619 628 630 631 632 635 636 640 645 654 657 661 662 667 674 680 680 682 684 688 692 694 697 702 704 705 705 706 707
 708 714 714 718 719 719 719 721 721 722 722 725 726 729 732 753 760 761 762 765 770 788 791 795 802 807 808 808 813 829 829 831 832 837 840 841 843 845 853 861 861 864
 867 870 871 873 878 881 884 885 888 904 904 909 916 918 925 927 927 928 935 936 946 948 951 960 965 968 985 994 996 997 1003 1005 1008 1009 1015 1017 1019 1019 1022

Process finished with exit code 0
```

```
/Library/Java/JavaVirtualMachines/temurin-17.jdk/Contents/Home/bin/java -javaagent:/Applications/IntelliJ IDEA.app/Contents/lib/idea_rt.jar=56732:/Applications/IntelliJ IDEA.app/Contents/bin -Dfile.encoding=UTF-8 -classpath /Users/lenisesilvio/IdeaProjects/ICT1018_Assignment/out/production/Question1 Question1
Array A before shell sort:
183 598 570 513 377 329 412 114 931 704 185 244 496 903 518 26 565 241 269 627 745 524 400 731 232 21 243 524 371 558 475 891 123 77 529 865 798 375 684 535 625 302 623
 166 640 40 623 670 402 65 249 294 945 945 690 852 616 897 510 959 206 314 165 849 963 992 948 353 633 30 417 256 822 962 689 914 87 106 926 160 407 383 627 749 401 355
 443 710 899 485 474 414 340 1021 186 729 471 693 937 488 849 356 896 405 568 290 776 159 61 375 922 905 908 85 5 942 594 328 435 921 639 1003 14 147 460 413 543 380 511
 629 632 734 366 370 340 63 961 1015 252 694 762 117 107 192 186 105 52 851 956 393 419 51 422 569 35 389 256 323 223 673 900 199 170 556 151 112 35 356 23 744 532 189
 343 653 26 429 615 356 395 452 920 78 147 766 830 376 416 348 439 536 255 432 428 853 711 440 150 646 245 367 900 159 984 414 26 231 175 952 476 750 656 743 940 958 394
 610 401 592 14 257 697 833 650 530 960 965 342 877 306 987 211 197 590 962 977 65 13 171 49 74 98 175 459 678 450 515 913 197 742 361 756 347 718 1001 390 814

Array A after shell sort:
5 13 14 14 21 23 26 26 26 30 35 35 40 49 51 52 61 63 65 65 74 77 78 85 87 98 105 106 107 112 114 117 123 147 147 150 151 159 159 160 165 166 170 171 175 175 183 185 186
 186 189 192 197 197 199 206 211 223 231 232 241 243 244 245 249 252 255 256 256 257 269 290 294 302 306 314 323 328 329 340 340 342 343 347 348 353 355 356 356 356 361
 366 367 370 371 375 375 376 377 380 383 389 390 393 394 395 400 401 401 402 405 407 412 413 414 414 416 417 419 422 428 429 432 435 439 440 443 450 452 459 460 471 474
 475 476 485 488 496 510 511 513 515 518 524 524 529 530 532 535 536 543 556 558 565 568 569 570 590 592 594 598 610 615 616 623 623 625 627 627 629 632 633 639 640 646
 650 653 656 670 673 678 684 689 690 693 694 697 704 710 711 718 729 731 734 742 743 744 745 749 750 756 762 766 776 798 814 822 830 833 849 849 851 852 853 865 877 891
 896 897 899 900 900 903 905 908 913 914 920 921 922 926 931 937 940 942 945 945 948 952 956 958 959 960 961 962 962 963 965 977 984 987 992 1001 1003 1015 1021


Array B before quick sort:
604 707 908 653 854 29 972 599 273 551 473 897 737 843 914 732 867 492 706 560 906 1008 76 351 760 638 676 457 398 520 897 1010 582 651 208 866 831 21 235 204 793 903 522
  626 898 924 249 711 273 553 779 341 82 422 846 481 33 748 767 460 315 832 320 955 785 655 520 378 479 507 90 29 509 662 953 616 117 108 218 368 213 454 411 897 69 685
 52 322 475 173 427 279 45 730 909 198 39 536 218 348 287 406 576 337 550 922 396 234 161 643 93 222 23 524 538 828 843 205 293 651 653 519 848 818 844 928 399 794 240
 108 751 913 742 685 711 369 1020 733 556 897 750 682 169 794 63 65 47 781 333 463 74 196 159 324 149 934 897 89 624 704 211 212 495 161 284 266 533 388 997 165 65 948
 580 423 744 556 694 430 900 381 562 889 971 756 634 170 548 507 435 973 342 959 824 664 898 275 426 116 83 659 587 316 627 780 396 67 64 27 227 923 959 275 482 916 464
 36 645 643 163 440 715 196 362 544 903 257 335 367 631 179 424 422 917 232 873 785 941 839 474 751 47 849 137 88 70 1001 129 859 808 277 187 614 943 518 468 490 297

Array B after quick sort:
21 23 27 29 29 33 36 39 45 47 47 52 63 64 65 65 67 69 70 74 76 82 83 88 89 90 93 108 108 116 117 129 137 149 159 161 161 163 165 169 170 173 179 187 196 196 198 204 205
 208 211 212 213 218 218 222 227 232 234 235 240 249 257 266 273 273 275 275 277 279 284 287 293 297 315 316 320 322 324 333 335 337 341 342 348 351 362 367 368 369 378
 381 388 396 396 398 399 406 411 422 422 423 424 426 427 430 435 440 454 457 460 463 464 468 473 474 475 479 481 482 490 492 495 507 507 509 518 519 520 520 522 524 533
 536 538 544 548 550 551 553 556 556 560 562 576 580 582 587 599 604 614 616 624 626 627 631 634 638 643 643 645 651 651 653 653 655 659 662 664 676 682 685 685 694 704
 706 707 711 711 715 730 732 733 737 742 744 748 750 751 751 756 760 767 779 780 781 785 785 793 794 794 808 818 824 828 831 832 839 843 843 844 846 848 849 854 859 866
 867 873 889 897 897 897 897 897 898 898 900 903 903 906 908 909 913 914 916 917 922 923 924 928 934 941 943 948 953 955 959 959 971 972 973 997 1001 1008 1010 1020

Process finished with exit code 0
```

Question 2

Time complexity: O(n)

The program builds on the program from question 1 but adds the `merge_sort` function that merges two arrays into one sorted array. The `main` function contains the test driver.

Program:

```
import java.util.Random;

public class Question2 {
    public static void main(String[] args) {
        /* Test Driver */
        int[] A = new int[256]; // first array
        random_array(256, A); // populate array with random numbers
        int[] B = new int[257]; // create second array
        random_array(257, B); // populate array with random numbers

        // Perform shell sort on array A
        shell_sort(A, 256);
        System.out.println("Array A:");
        print_array(A, 256);

        System.out.println("\n");

        // Perform quick sort on array B
        quick_sort(B, 0, 256);
        System.out.println("Array B:");
        print_array(B, 257);

        System.out.println("\n");

        int[] C = new int[256+257]; // create new array to store merged
array
        merge_sort(A, B, C); // merga array A and B
        System.out.println("Array C: ");
        print_array(C, 256+257); // display merged array

    }

    //prints array
    static void print_array(int arr[], int size) {
        for (int i = 0; i < size; i++) {
            System.out.print(arr[i] + " ");
        }
        System.out.println();
    }

    // Populates an array with random numbers from 0 to 1024
    static void random_array(int n, int arr[]) {
        Random rn = new Random();

        // generate random number for each index in the array
        for (int i = 0; i < n; i++) {
            arr[i] = rn.nextInt(0, 1024);
        }
```

```
    }

    //Shell sort algorithm
    // Adapted from: https://www.geeksforgeeks.org/shellsort/
    static void shell_sort(int A[], int size) {
        int gap = size / 2; // sets initial gap size to half the array
length

        // while there is a gap of at least 1 between each element
        while (gap > 0) {
            // perform insertion sort but with the specified gap
            for (int i = gap;  i < size; i++) {
                int j = i;
                int temp = A[i];
                // find correct position for A[i]
                // compare elements at the specified gap
                // shift elements larger than temp down the array by the
specified gap
                while (j >= gap) {
                    if (temp > A[j-gap]) { // break if temp is in its
correct position
                        break;
                    }
                    A[j] = A[j-gap];
                    j -= gap;
                }
                A[j] = temp; // put element a[i] to its correct position
            }
            gap = gap / 2; // half the gap after each iteration
        }
    }

    // Quick sort algorithm
    // Adapted from: https://www.geeksforgeeks.org/quicksort-using-random-
pivoting/
    static void quick_sort(int array[], int start, int end) {
        if (start < end) { //continuously sort array while partitioned
array is larger than one
            Random rn = new Random();
            swap(array, rn.nextInt(end - start) + start, end); // choose
random number as pivot and put it lass
            int p = partition(array, start, end); // call partition
function
            quick_sort(array, start, p - 1); //recursively call quick_sort
function on left part of previously partitioned array
            quick_sort(array, p + 1, end); //recursively call quick_sort
function on right part of previously partitioned array
        }
    }

    // Partition function for quick sort
    static int partition(int array[], int start, int end) {
        int pivot = array[end]; // find the pivot

        int x = start; //set to the starting index of the partition

        for (int i = start; i < end; i++) {
            if (array[i] <= pivot) { // put elements smaller than or equal
to the pivot to the left
                swap(array, i, x);
                x++;
```

```
            }
        }
        swap(array, x, end); //put pivot to its correct place
        return x;
    }


    // swaps to elements in an array
    static void swap(int array[], int index1, int index2) {
        int temp = array[index1]; // save index temporarily for swap
        array[index1] = array[index2]; //set element at index1 equal to
element at index2
        array[index2] = temp; //set element at index 2 as being equal to
the temp variable
    }


    // Merge to sort array into one sorted array
    static void merge_sort (int array_1[], int array_2[], int
merged_array[]) {
        int i = 0, j = 0, k = 0; // keeps track of the indexes

        // adds elements to the merged_array until no elements are left
from either array_1 or array_2
        while (i < array_1.length && j < array_2.length) {
            if (array_1[i] <= array_2[j]) { // add element of array_1 if it
is smaller or equal to the element in array_2
                merged_array[k] = array_1[i];
                i++; // increment the index of array_1
            } else { // add element from array_2
                merged_array[k] = array_2[j];
                j++; // increment the index of array_2
            }
            k++; // increment index of merged_array
        }

        // add any remaining elements from array_1
        for (int l = i; l < array_1.length; l++, k++) {
            merged_array[k] = array_1[l];
        }

        // add any remaining elements from array_2
        for (int l = j; l < array_2.length; l++, k++) {
            merged_array[k] = array_2[l];
        }
    }
}
```

Outputs:

```
/Library/Java/JavaVirtualMachines/temurin-17.jdk/Contents/Home/bin/java -javaagent:/Applications/IntelliJ IDEA.app/Contents/lib/idea_rt.jar=56896:/Applications/IntelliJ IDEA.app/Contents/bin -Dfile.encoding=UTF-8 -classpath /Users/lenisesilvio/IdeaProjects/ICT1018_Assignment/out/production/Question2 Question2
Array A:
2 3 3 7 11 12 13 14 17 26 51 54 57 62 63 66 71 73 75 77 77 78 84 93 93 94 98 105 111 112 116 117 118 122 123 126 127 128 130 143 146 149 152 155 162 165 165 170 171 175
 181 184 191 206 217 221 223 225 234 240 250 250 257 257 258 273 278 278 288 296 296 298 305 306 306 312 312 316 316 317 320 320 321 323 335 341 342 345 350 350 360 361
 367 369 376 378 379 380 381 382 389 391 396 412 412 415 424 424 424 425 426 428 431 442 448 449 454 465 469 472 479 483 484 485 487 488 490 491 497 498 512 514 524 527
 536 540 540 550 556 560 566 567 567 567 569 580 588 588 588 594 604 613 615 617 624 624 625 628 630 630 634 635 637 640 642 649 658 659 660 673 676 678 683 683 687 688
 698 699 701 705 706 706 706 710 712 716 717 722 726 730 733 734 739 741 744 749 749 750 752 758 760 763 765 767 768 768 773 775 778 779 789 791 798 806 824 825 833 836
 849 849 850 850 858 864 867 875 876 883 889 890 893 903 904 913 914 926 933 944 944 949 949 962 966 968 968 974 975 978 982 986 988 988 996 1004 1005 1020


Array B:
0 0 4 8 16 16 17 22 34 36 42 44 45 49 53 60 73 84 85 85 91 92 95 97 97 98 100 107 107 126 127 132 135 141 144 145 154 159 160 160 165 170 177 178 185 198 201 205 217 221
 226 232 234 239 242 246 247 252 263 270 270 273 274 275 284 285 288 289 301 302 317 317 319 321 322 331 334 339 342 342 346 355 358 361 361 364 364 368 370 371 377 381
 384 385 387 387 387 388 388 392 397 397 398 419 423 429 430 436 440 444 446 448 450 452 454 455 460 465 467 467 478 482 487 488 492 495 498 499 511 521 533 538 541 545
 547 554 558 558 565 568 569 571 572 573 574 582 585 585 586 595 598 599 601 621 622 632 633 634 635 645 645 648 650 651 655 656 656 667 683 683 686 692 700 701 713 721
 721 721 725 725 726 731 732 732 739 745 764 764 769 775 775 776 778 778 782 782 787 789 790 793 798 805 806 807 812 812 817 834 840 860 870 879 883 886 888 890 895 896
 901 907 908 914 914 916 923 927 927 928 933 936 940 941 944 954 955 957 958 958 962 962 967 978 978 986 986 989 993 995 997 998 999 999 1006 1013 1014 1014 1017


Array C:
0 0 2 3 3 4 7 8 11 12 13 14 16 16 17 17 22 26 34 36 42 44 45 49 51 53 54 57 60 62 63 66 71 73 73 75 77 77 78 84 84 85 85 91 92 93 93 94 95 97 97 98 98 100 105 107 107 111
  112 116 117 118 122 123 126 126 127 127 128 130 132 135 141 143 144 145 146 149 152 154 155 159 160 160 162 165 165 165 170 170 171 175 177 178 181 184 185 191 198 201
 205 206 217 217 221 221 223 225 226 232 234 234 239 240 242 246 247 250 250 252 257 257 258 263 270 270 273 273 274 275 278 278 284 285 288 288 289 296 296 298 301 302
 305 306 306 312 312 316 316 317 317 317 319 320 320 321 321 322 323 331 334 335 339 341 342 342 342 345 346 350 350 355 358 360 361 361 361 364 364 367 368 369 370 371
 376 377 378 379 380 381 381 382 384 385 387 387 387 388 388 389 391 392 396 397 397 398 412 412 415 419 423 424 424 424 425 426 428 429 430 431 436 440 442 444 446 448
 448 449 450 452 454 454 455 460 465 465 467 467 469 472 478 479 482 483 484 485 487 487 488 488 490 491 492 495 497 498 498 499 511 512 514 521 524 527 533 536 538 540
 540 541 545 547 550 554 556 558 558 560 565 566 567 567 567 568 569 569 571 572 573 574 580 582 585 585 586 588 588 588 594 595 598 599 601 604 613 615 617 621 622 624
 624 625 628 630 630 632 633 634 634 635 635 637 640 642 645 645 648 649 650 651 655 656 656 658 659 660 667 673 676 678 683 683 683 683 686 687 688 692 698 699 700 701
 701 705 706 706 706 710 712 713 716 717 721 721 721 722 725 725 726 726 730 731 732 732 733 734 739 739 741 744 745 749 749 750 752 758 760 763 764 764 765 767 768 768
 769 773 775 775 775 776 778 778 778 779 782 782 787 789 789 790 791 793 798 798 805 806 806 807 812 812 817 824 825 833 834 836 840 849 849 850 850 858 860 864 867 870
 875 876 879 883 883 886 888 889 890 890 893 895 896 901 903 904 907 908 913 914 914 914 916 923 926 927 927 928 933 933 936 940 941 944 944 944 949 949 954 955 957 958
 958 962 962 962 966 967 968 968 974 975 978 978 978 982 986 986 986 988 988 989 993 995 996 997 998 999 999 1004 1005 1006 1013 1014 1014 1017 1020

Process finished with exit code 0



/Library/Java/JavaVirtualMachines/temurin-17.jdk/Contents/Home/bin/java -javaagent:/Applications/IntelliJ IDEA.app/Contents/lib/idea_rt.jar=56915:/Applications/IntelliJ IDEA.app/Contents/bin -Dfile.encoding=UTF-8 -classpath /Users/lenisesilvio/IdeaProjects/ICT1018_Assignment/out/production/Question2 Question2
Array A:
1 3 6 10 10 11 13 15 17 22 23 28 34 36 38 43 44 46 47 47 53 54 66 75 80 82 84 87 87 92 95 97 99 107 111 115 116 122 139 144 152 152 153 164 164 168 170 171 171 174 178
 179 188 188 193 195 200 202 205 207 209 209 214 224 235 253 253 254 258 263 264 265 265 266 267 275 281 284 285 286 298 308 312 315 331 335 349 353 356 357 358 362 369
 370 372 376 377 381 383 391 391 400 401 402 402 415 416 440 449 454 454 456 459 471 474 475 497 497 500 501 504 504 510 513 515 516 518 521 524 528 533 539 542 543 551
 554 559 560 564 574 576 582 583 583 588 593 595 598 598 621 624 625 632 647 653 656 658 662 663 670 682 682 686 686 689 696 704 707 713 714 714 725 733 737 742 743 743
 748 755 762 766 766 767 771 779 783 791 807 808 808 809 821 822 829 830 830 831 832 834 838 838 839 843 848 850 856 857 857 867 875 879 881 882 890 891 897 899 900 905
 909 912 912 914 925 931 932 941 943 947 948 951 952 954 956 957 962 968 970 971 972 977 979 982 985 989 992 999 1000 1006 1006 1006 1008 1009 1010 1014 1014


Array B:
0 4 5 9 12 18 30 37 46 57 59 66 67 69 75 79 83 84 89 93 99 106 108 112 116 118 120 129 129 136 140 141 143 145 145 161 167 171 174 179 181 184 184 189 192 195 201 205 211
  217 217 218 219 220 225 226 227 234 240 244 244 248 249 258 267 274 281 287 288 288 292 300 305 307 313 314 319 328 335 335 341 342 343 347 348 348 348 349 359 364 368
 377 379 381 384 388 399 402 416 416 416 422 427 434 440 442 443 446 458 463 472 475 475 484 486 489 491 492 494 496 497 503 511 513 514 517 527 530 538 539 542 542 550
 554 556 560 562 565 566 571 571 580 582 583 584 589 589 593 602 602 603 608 612 618 621 621 622 624 634 635 637 643 651 661 666 671 673 676 676 683 688 691 695 696 702
 708 710 722 724 726 726 731 737 739 743 746 747 749 751 751 755 757 757 762 768 770 771 774 783 790 819 822 825 827 830 838 841 849 849 857 863 868 892 896 897 898 901
 902 907 907 911 912 912 915 918 920 933 934 935 941 946 946 949 949 951 954 954 959 961 967 977 980 981 983 983 984 990 996 997 1010 1011 1012 1013 1016 1018 1019 1020


Array C:
0 1 3 4 5 6 9 10 10 11 12 13 15 17 18 22 23 28 30 34 36 37 38 43 44 46 46 47 47 53 54 57 59 66 66 67 69 75 75 79 80 82 83 84 84 87 87 89 92 93 95 97 99 99 106 107 108 111
  112 115 116 116 118 120 122 129 129 136 139 140 141 143 144 145 145 152 152 153 161 164 164 167 168 170 171 171 171 174 174 178 179 179 181 184 184 188 188 189 192 193
 195 195 200 201 202 205 205 207 209 209 211 214 217 217 218 219 220 224 225 226 227 234 235 240 244 244 248 249 253 253 254 258 258 263 264 265 265 266 267 267 274 275
 281 281 284 285 286 287 288 288 292 298 300 305 307 308 312 313 314 315 319 328 331 335 335 335 341 342 343 347 348 348 348 349 349 353 356 357 358 359 362 364 368 369
 370 372 376 377 377 379 381 381 383 384 388 391 391 399 400 401 402 402 402 415 416 416 416 416 422 427 434 440 440 442 443 446 449 454 454 456 458 459 463 471 472 474
 475 475 475 484 486 489 491 492 494 496 497 497 497 500 501 503 504 504 510 511 513 513 514 515 516 517 518 521 524 527 528 530 533 538 539 539 542 542 542 543 550 551
 554 554 556 559 560 560 562 564 565 566 571 571 574 576 580 582 582 583 583 583 584 588 589 589 593 593 595 598 598 602 602 603 608 612 618 621 621 621 622 624 624 625
 632 634 635 637 643 647 651 653 656 658 661 662 663 666 670 671 673 676 676 682 682 683 686 686 688 689 691 695 696 696 702 704 707 708 710 713 714 714 722 724 725 726
 726 731 733 737 737 739 742 743 743 743 746 747 748 749 751 751 755 755 757 757 762 762 766 766 767 768 770 771 771 774 779 783 783 790 791 807 808 808 809 819 821 822
 822 825 827 829 830 830 830 831 832 834 838 838 838 839 841 843 848 849 849 850 856 857 857 857 863 867 868 875 879 881 882 890 891 892 896 897 897 898 899 900 901 902
 905 907 907 909 911 912 912 912 912 914 915 918 920 925 931 932 933 934 935 941 941 943 946 946 947 948 949 949 951 951 952 954 954 954 956 957 959 961 962 967 968 970
 971 972 977 977 979 980 981 982 983 983 984 985 989 990 992 996 997 999 1000 1006 1006 1006 1008 1009 1010 1010 1011 1012 1013 1014 1014 1016 1018 1019 1020

Process finished with exit code 0
```

```
/Library/Java/JavaVirtualMachines/temurin-17.jdk/Contents/Home/bin/java -javaagent:/Applications/IntelliJ IDEA.app/Contents/lib/idea_rt.jar=56917:/Applications/IntelliJ IDEA.app/Contents/bin -Dfile.encoding=UTF-8 -classpath /Users/lenisesilvio/IdeaProjects/ICT1018_Assignment/out/production/Question2 Question2
Array A:
5 11 14 18 20 21 25 31 33 35 44 60 62 66 71 77 77 78 84 91 92 92 93 97 97 105 110 111 117 119 122 125 127 128 136 140 144 148 150 153 169 175 180 180 181 186 188 192 193
 205 207 213 214 220 227 229 237 241 242 267 270 272 273 277 278 280 289 296 299 300 309 310 311 321 324 325 333 337 337 337 343 344 349 350 350 360 364 365 367 372 381
 384 391 392 394 395 395 400 400 401 402 402 409 409 410 411 413 417 424 426 432 434 438 438 440 443 445 451 452 454 482 486 489 490 490 492 499 500 502 505 507 508 511
 522 522 523 533 534 535 544 554 556 556 564 572 579 582 593 594 594 597 601 604 604 605 612 612 616 626 629 637 641 646 648 650 651 661 661 661 661 662 668 676 684
 690 699 703 703 716 717 724 736 736 737 737 738 742 763 765 771 773 776 791 801 803 815 823 826 830 832 834 840 845 846 854 857 857 859 862 864 866 866 867 867 881 881
 890 890 892 895 896 898 899 899 913 916 923 926 931 936 936 944 945 952 953 953 958 960 961 965 965 965 966 966 983 984 984 990 997 1000 1001 1002 1009 1010 1015


Array B:
6 7 11 14 18 22 27 28 30 37 39 47 51 52 52 57 60 65 65 69 72 76 95 101 103 108 108 109 114 118 121 122 125 129 129 140 141 147 147 151 154 155 158 159 163 165 169 174 191
  196 197 200 202 203 207 211 213 228 228 233 233 238 241 243 252 254 256 259 261 263 265 271 283 284 291 295 298 298 302 302 302 306 311 313 315 317 322 322 325 325 328
 328 330 337 339 340 347 352 357 367 375 377 397 400 407 413 414 419 422 427 428 430 443 446 447 452 461 469 470 471 471 477 478 480 486 489 495 496 497 502 503 504 508
 515 523 524 527 528 530 533 538 539 540 546 547 548 552 559 561 561 568 579 585 589 591 594 601 603 603 603 603 606 608 614 617 618 624 625 626 637 639 644 649 651 655
 657 659 669 672 672 675 680 685 690 695 696 698 701 703 704 709 714 717 720 720 723 723 725 734 738 738 743 745 745 745 748 761 768 768 779 780 783 784 790 790 797 804
 825 832 836 839 846 865 872 879 882 886 886 889 894 900 903 904 925 931 935 945 948 950 952 955 962 969 971 973 974 978 980 980 986 991 992 1000 1004 1005 1016 1023


Array C:
5 6 7 11 11 14 14 18 18 20 21 22 25 27 28 30 31 33 35 37 39 44 47 51 52 52 57 60 60 62 65 65 66 69 71 72 76 77 77 78 84 91 92 92 93 95 97 97 101 103 105 108 108 109 110
 111 114 117 118 119 121 122 122 125 125 127 128 129 129 136 140 140 141 144 147 147 148 150 151 153 154 155 158 159 163 165 169 169 174 175 180 180 181 186 188 191 192
 193 196 197 200 202 203 205 207 207 211 213 213 214 220 227 228 228 229 233 233 237 238 241 241 242 243 252 254 256 259 261 263 265 267 270 271 272 273 277 278 280 283
 284 289 291 295 296 298 298 299 300 302 302 302 306 309 310 311 311 313 315 317 321 322 322 324 325 325 325 328 328 330 333 337 337 337 337 339 340 343 344 347 349 350
 350 352 357 360 364 365 367 367 372 375 377 381 384 391 392 394 395 395 397 400 400 400 401 402 402 407 409 409 410 411 413 413 414 417 419 422 424 426 427 428 430 432
 434 438 438 440 443 443 445 446 447 451 452 452 454 461 469 470 471 471 477 478 480 482 486 486 489 489 490 490 492 495 496 497 499 500 502 502 503 504 505 507 508 508
 511 515 522 522 523 523 524 527 528 530 533 533 533 534 535 538 539 540 544 546 547 548 552 554 556 556 559 561 561 564 568 572 579 579 582 585 589 591 593 594 594 594
 597 601 601 603 603 603 603 604 604 605 606 608 612 612 614 616 617 618 624 625 626 626 629 637 637 639 641 644 646 648 649 650 651 651 655 657 659 661 661 661 661 662
 668 669 672 672 675 676 680 684 685 690 690 695 696 698 699 701 703 703 703 704 709 714 716 717 717 720 720 723 723 724 725 734 736 736 737 737 738 738 738 742 743 745
 745 745 748 761 763 765 768 768 771 773 776 779 780 783 784 790 790 791 797 801 803 804 815 823 825 826 830 832 832 834 836 839 840 845 846 846 854 857 857 859 862 864
 865 866 866 867 867 872 879 881 881 882 886 886 889 890 890 892 894 895 896 898 899 899 900 903 904 913 916 923 925 926 931 931 935 936 936 944 945 945 948 950 952 952
 953 953 955 958 960 961 962 965 965 965 966 966 969 971 973 974 978 980 980 983 984 984 986 990 991 992 997 1000 1000 1001 1002 1004 1005 1009 1010 1015 1016 1023

Process finished with exit code 0
```

```
/Library/Java/JavaVirtualMachines/temurin-17.jdk/Contents/Home/bin/java -javaagent:/Applications/IntelliJ IDEA.app/Contents/lib/idea_rt.jar=56920:/Applications/Intel
 IDEA.app/Contents/bin -Dfile.encoding=UTF-8 -classpath /Users/lenisesilvio/IdeaProjects/ICT1018_Assignment/out/production/Question2 Question2
Array A:
4 6 12 17 18 19 24 28 32 32 37 42 44 48 48 54 64 69 75 76 77 79 81 83 83 85 88 94 102 103 104 105 107 108 115 116 117 122 136 142 153 154 158 159 160 167 170 176 180 184
 184 188 194 203 206 206 207 210 213 217 218 219 221 221 222 223 224 235 236 238 240 243 252 253 259 260 261 261 264 270 271 272 273 275 278 279 281 281 285 292 298 308
 316 332 344 351 354 356 360 371 374 376 391 393 399 405 405 406 407 410 416 416 421 423 424 425 427 431 440 446 447 451 454 455 455 458 459 469 472 477 482 486 487 490
 491 500 504 506 510 530 530 536 544 545 552 556 557 559 561 570 572 573 579 591 603 606 614 616 620 621 623 627 639 644 644 646 648 650 652 658 660 663 664 665 676 678
 682 686 687 691 700 708 708 722 727 747 757 765 767 768 769 774 781 788 788 791 801 803 812 813 824 831 831 831 834 835 836 837 839 841 842 843 845 850 851 853 859 863
 876 877 881 882 892 899 903 911 912 916 919 922 931 941 942 957 958 958 959 965 966 969 969 970 973 973 981 996 1001 1002 1002 1004 1006 1009 1010 1011 1017 1023


Array B:
3 4 5 11 12 17 24 24 29 31 46 59 67 91 94 98 115 127 132 136 138 142 148 148 152 161 165 172 179 180 188 192 196 204 207 207 211 219 225 230 240 250 251 253 256 258 261
 261 266 275 275 282 285 285 288 294 298 309 315 322 324 324 327 329 333 336 338 340 343 345 353 358 366 378 380 380 385 389 403 405 405 407 408 409 410 419 419 421 424
 434 443 443 449 454 458 465 467 476 479 479 480 492 494 497 500 505 508 515 518 526 527 528 531 533 535 538 539 546 563 565 567 574 578 578 582 588 593 593 594 597 600
 602 603 605 606 607 609 609 611 611 616 616 617 623 623 625 625 632 636 637 644 648 654 659 664 664 665 675 675 682 682 687 691 696 696 702 721 733 740 743 743 745 745
 751 752 752 758 760 764 765 769 771 783 784 785 789 790 794 798 799 802 802 802 804 804 806 817 819 823 830 832 833 840 846 848 853 853 855 856 859 863 871 871 873 873
 887 887 889 891 893 895 900 914 915 919 920 922 925 930 931 935 937 937 943 947 950 952 954 957 965 966 971 982 983 983 984 987 987 992 997 998 1005 1005 1009 1011 1013
 1018


Array C:
3 4 4 5 6 11 12 12 17 17 18 19 24 24 24 28 29 31 32 32 37 42 44 46 48 48 54 59 64 67 69 75 76 77 79 81 83 83 85 88 91 94 94 98 102 103 104 105 107 108 115 115 116 117 122
  127 132 136 136 138 142 142 148 148 152 153 154 158 159 160 161 165 167 170 172 176 179 180 180 184 184 188 188 192 194 196 203 204 206 206 207 207 207 210 211 213 217
 218 219 219 221 221 222 223 224 225 230 235 236 238 240 240 243 250 251 252 253 253 256 258 259 260 261 261 261 261 264 266 270 271 272 273 275 275 275 278 279 281 281
 282 285 285 285 288 292 294 298 298 308 309 315 316 322 324 324 327 329 332 333 336 338 340 343 344 345 351 353 354 356 358 360 366 371 374 376 378 380 380 385 389 391
 393 399 403 405 405 405 405 406 407 407 408 409 410 410 416 416 419 419 421 421 423 424 424 425 427 431 434 440 443 443 446 447 449 451 454 454 455 455 458 458 459 465
 467 469 472 476 477 479 479 480 482 486 487 490 491 492 494 497 500 500 504 505 506 508 510 515 518 526 527 528 530 530 531 533 535 536 538 539 544 545 546 552 556 557
 559 561 563 565 567 570 572 573 574 578 578 579 582 588 591 593 593 594 597 600 602 603 603 605 606 606 607 609 609 611 611 614 616 616 616 617 620 621 623 623 623 625
 625 627 632 636 637 639 644 644 644 646 648 648 650 652 654 658 659 660 663 664 664 664 665 665 675 675 676 678 682 682 682 686 687 687 691 691 696 696 700 702 708 708
 721 722 727 733 740 743 743 745 745 747 751 752 752 757 758 760 764 765 765 767 768 769 769 771 774 781 783 784 785 788 788 789 790 791 794 798 799 801 802 802 802 803
 804 804 806 812 813 817 819 823 824 830 831 831 831 832 833 834 835 836 837 839 840 841 842 843 845 846 848 850 851 853 853 853 855 856 859 859 863 863 871 871 873 873
 876 877 881 882 887 887 889 891 892 893 895 899 900 903 911 912 914 915 916 919 919 920 922 922 925 930 931 931 935 937 937 941 942 943 947 950 952 954 957 957 958 958
 959 965 965 966 966 969 969 970 971 973 973 981 982 983 983 984 987 987 992 996 997 998 1001 1002 1002 1004 1005 1005 1006 1009 1009 1010 1011 1011 1013 1017 1018 1023

Process finished with exit code 0
```

Question 3

Time complexity: O(n)

The `extreme_points` function checks whether an array has any extreme points. The test driver is the `main` function. An array has no extreme points only when it is sorted in ascending order since the only way for an array to be sorted in ascending order is for all elements (apart from the first and last element) to fulfil the following condition: `A[i-1]` ≤ `A[i]` ≤ `A[i+1]` i.e. the elements of the array do not fulfil the conditions to be extreme points.

Program:

```
public class Question3 {
    public static void main(String[] args) {
        /* Test driver */
        int array1[] = {5, 3, 1, 2, 9};
        System.out.println("Checking array1...");
        extreme_points(array1);
        System.out.println();

        int array2[] = {1, 2, 3, 4, 5, 6, 7};
        System.out.println("Checking array2...");
        extreme_points(array2);
        System.out.println();

        int array3[] = {1, 2, 3, 5, 4};
        System.out.println("Checking array3...");
        extreme_points(array3);
        System.out.println();

        int array4[] = {9, 2, 4, 6, 3};
        System.out.println("Checking array4...");
        extreme_points(array4);
        System.out.println();

        int array5[] = {5, 4, 3, 2, 1};
        System.out.println("Checking array5...");
        extreme_points(array5);
        System.out.println();
    }

    // Checks array for extreme points
    static void extreme_points(int array[]) {
        boolean sorted = true; // keeps track of whether array has any
extreme points or not
        // check all elements in the array except the first and last
        for (int i = 1; i < array.length-1; i++) {
            // compare current element with previous and subsequent element
            if (array[i-1] > array[i] || array[i] > array[i+1]) {
                // if element before it is larger or element after it is
smaller
                sorted = false; // set sorted to false
                System.out.println(array[i]); //print element
            }
        }
```

```
            // if sorted is true
            if (sorted) {
                System.out.println("SORTED");
            }
        }
}
```

Output:

```
/Library/Java/JavaVirtualMachines/temurin-17.jdk/Contents/Home/bin/java -javaagent:/Applications/IntelliJ IDEA.app/Contents/lib/idea_rt.jar=56997:/Applications/IntelliJ
 IDEA.app/Contents/bin -Dfile.encoding=UTF-8 -classpath /Users/lenisesilvio/IdeaProjects/ICT1018_Assignment/out/production/question3 Question3
Checking array1...
3
1

Checking array2...
SORTED

Checking array3...
5

Checking array4...
2
6

Checking array5...
4
3
2


Process finished with exit code 0
```

Question 4

Time complexity: $O(n^2)$

The function `find_multiples` finds all the pairs of numbers which multiply to the same product. The product is the key and all pairs of numbers which multiply to the product are stored in an array. The function `print_multiples` is used to display the pairs for testing.

Program:

```
import java.util.ArrayList;
import java.util.HashMap;

public class Question4 {
    public static void main(String[] args) {
        /* Test Driver */
        int array1[] = {1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 12};
        HashMap<Integer, ArrayList<Integer>> product_multiples1 = new
HashMap<>();
        find_multiples(array1, product_multiples1);
        System.out.println("Multiples in array1...");
        print_multiples(product_multiples1);
        System.out.println("\n");

        int array2[] = {2, 2, 3, 4, 4, 5, 6};
        HashMap<Integer, ArrayList<Integer>> product_multiples2 = new
HashMap<>();
        find_multiples(array2, product_multiples2);
        System.out.println("Multiples in array2...");
        print_multiples(product_multiples2);
        System.out.println("\n");

        int array3[] = {124, 893, 796, 217, 922, 1013, 156, 135, 544, 55,
81, 381, 526, 86, 886, 779, 296, 296};
        HashMap<Integer, ArrayList<Integer>> product_multiples3 = new
HashMap<>();
        find_multiples(array3, product_multiples2);
        System.out.println("Multiples in array3...");
        print_multiples(product_multiples3);
        System.out.println("\n");

    }

    // Prints multiples
    static void print_multiples(HashMap<Integer, ArrayList<Integer>>
multiples) {
        // go through all the key-value pairs in the hashmap
        for (int multiple : multiples.keySet()) {
            ArrayList<Integer> multiple_pairs =  multiples.get(multiple);
            // if there are at least to pairs of numbers which multiply to
the same number
            // print all pair combinations
            if (multiple_pairs.size() >= 4) {
                System.out.println("Multiple pairs for " + multiple + ":");
                for (int i = 0; i < multiple_pairs.size()-2; i+=2) {
                    for (int j = i+2 ; j < multiple_pairs.size(); j+=2) {
                        System.out.println(multiple_pairs.get(i) + " * " +
```

```
multiple_pairs.get(i+1) + " = " + multiple_pairs.get(j) + " * " +
multiple_pairs.get(j+1));
                    }
                }
            }
        }
    }

    // stores all products and their multiples in a hashmap
    // key is product
    // values is an array of all its products
    static void find_multiples(int array[], HashMap<Integer,
ArrayList<Integer>> multiples) {
        // multiply every element in the array with each other to find the
products
        for (int i = 0; i < array.length-1; i++) {
            for (int j = i+1; j < array.length; j++) {
                int product = array[i] * array[j];
                // if product is already in the hashmap
                if (multiples.containsKey(product)) {
                    // skip any repeated multiples that are already added
                    if (multiples.get(product).contains(array[i]) ||
multiples.get(product).contains(array[j])) {
                        continue;
                    }
                    multiples.get(product).add(array[i]);
                    multiples.get(product).add(array[j]);
                } else { // add new product and multiples to hashmap
                    ArrayList<Integer> multiple_pair = new ArrayList<>();
                    multiple_pair.add(array[i]);
                    multiple_pair.add(array[j]);
                    multiples.put(product, multiple_pair);
                }
            }
        }
    }
}
```

Output:

```
/Library/Java/JavaVirtualMachines/temurin-17.jdk/Contents/Home/bin/java -javaagent:/Applications/IntelliJ IDEA.app/Contents/lib/idea_rt.jar=58230:/Applications/IntelliJ IDEA.app/Contents/bin -Dfile.encoding=UTF-8 -classpath /Users/lenisesilvio/IdeaProjects/ICT1018_Assignment/out/production/Question4 Question4
Multiples in array1...
Multiple pairs for 6:
1 * 6 = 2 * 3
Multiple pairs for 8:
1 * 8 = 2 * 4
Multiple pairs for 72:
6 * 12 = 8 * 9
Multiple pairs for 10:
1 * 10 = 2 * 5
Multiple pairs for 12:
1 * 12 = 2 * 6
1 * 12 = 3 * 4
2 * 6 = 3 * 4
Multiple pairs for 18:
2 * 9 = 3 * 6
Multiple pairs for 20:
2 * 10 = 4 * 5
Multiple pairs for 24:
2 * 12 = 3 * 8
2 * 12 = 4 * 6
3 * 8 = 4 * 6
Multiple pairs for 30:
3 * 10 = 5 * 6
Multiple pairs for 36:
3 * 12 = 4 * 9
Multiple pairs for 40:
4 * 10 = 5 * 8
Multiple pairs for 48:
4 * 12 = 6 * 8
Multiple pairs for 60:
5 * 12 = 6 * 10


Multiples in array2...
Multiple pairs for 12:
2 * 6 = 3 * 4


Multiples in array3...



Process finished with exit code 0
```

Question 5

Time complexity: O(n)

The stack is implemented using a linked list. The `main` contains the program that performs the RPN evaluation. The RPN expression is evaluated by parsing the expression and either adding the input to the stack if it is an integer or performing the corresponding operation if the input is an operand. The program could be improved by checking that the stack is not empty before popping, checking whether the user input is an integer or not to reduce some of the repetitive code, and checking if the user input is using any operators that are not allowed and checking whether the expression is separated by commas or not.

Program:

Node:

```
// Create node for linked list
public class Node {
    private int number;
    private Node previous;

    // Node constructor
    public Node(int number) {
        this.number = number;
        this.previous = null;
    }

    // Node constructor
    public Node(int number, Node previous) {
        this.number = number;
        this.previous = previous;
    }

    // Get previous node
    public Node getPrevious() {
        return this.previous;
    }

    // Get item stored in node
    public int getNumber() {
        return this.number;
    }
}
```

Stack:

```
// Create stack
public class Stack {
    private Node head; // store head of stack

    // stack constructor
    public Stack () {
```

```
        this.head = null;
    }

    // Pop topmost item in stack
    public int pop () {
        int popped = head.getNumber();
        head = head.getPrevious();
        return popped;
    }

    // Add item to stack
    public void push(int number) {
        head = new Node(number, this.head);
    }

    // Print contents of the stack
    public void print_stack() {
        System.out.println("Stack contents:");
        Node node = head;
        while (node != null) {
            System.out.println(node.getNumber());
            node = node.getPrevious();
        }
    }

    // Find amount of items in stack
    public int stack_length() {
        Node node = head;
        int total = 0;
        while (node != null) {
            total ++;
            node = node.getPrevious();
        }
        return total;
    }

    // check if stack is empty
    public void empty() {
        while (this.stack_length() != 0) {
            this.pop();
        }
    }
}
```

RPN evaluation:

```
import java.util.Scanner;

public class Question5 {
    public static void main(String[] args) {
        /* Test driver and RPN evaluation */
        Stack stack = new Stack(); // create new stack object
        Scanner scanner = new Scanner(System.in); // create scanner

        // Get user input and perform RPN evaluation
        while (true) {
            System.out.println("Enter RPN expression to evaluate with
values and operands seperated by a comma");
```

```
            System.out.println("Enter 'quit' to exit");
            String input = scanner.nextLine(); // get input from user
            if (input.equals("quit")) { // exit loop
                break;
            }
            String[] RPNexpression = input.split(","); // parse user input
using comma
            for(String value: RPNexpression) { // go through parsed input
and perform required action
                // if input is an operator
                switch (value) {
                    case "x" -> { // perform multiplication
                        // pop the two topmost elements in the stack
                        int a = stack.pop();
                        int b = stack.pop();
                        stack.push(a * b); // add result back onto stack

                    }
                    case "+" -> { // perform addition
                        // pop the two topmost elements in the stack
                        int a = stack.pop();
                        int b = stack.pop();
                        stack.push(a + b); // add result back onto stack

                    }
                    case "-" -> { // perform subtraction
                        // pop the two topmost elements in the stack
                        int a = stack.pop();
                        int b = stack.pop();
                        stack.push(a - b); // add result back onto stack

                    }
                    case "/" -> { // perform division
                        // pop the two topmost elements in the stack
                        int a = stack.pop();
                        int b = stack.pop();
                        stack.push(b / a); // add result back onto stack

                    }
                    default ->  // if input is an operand
                            stack.push(Integer.valueOf(value)); // add
operand to stack
                }
                stack.print_stack(); // print contents of the stack
            }
            stack.empty();
        }
    }
}
```

Output:

```
/Library/Java/JavaVirtualMachines/temurin-17.jdk/Contents/Home/bin/java -javaagent:/Applications/IntelliJ IDEA.app/Contents/lib/idea_rt.jar=58359:/Applications/IntelliJ IDEA.app/Contents/bin -Dfile.encoding=UTF-8 -classpath /Users/lenisesilvio/IdeaProjects/ICT1018 Assignment/out/production/Question5 Question5
Enter RPN expression to evaluate with values and operands seperated by a comma
Enter 'quit' to exit
3,5,2,x,+
Stack contents:
3
Stack contents:
5
3
Stack contents:
2
5
3
Stack contents:
10
3
Stack contents:
13
Enter RPN expression to evaluate with values and operands seperated by a comma
Enter 'quit' to exit
2,1,+,3,x
Stack contents:
2
Stack contents:
1
2
Stack contents:
3
Stack contents:
3
3
Stack contents:
9
Enter RPN expression to evaluate with values and operands seperated by a comma
Enter 'quit' to exit
4,13,5,/,+
Stack contents:
4
Stack contents:
13
4
Stack contents:
5
13
4
Stack contents:
2
4
Stack contents:
6
Enter RPN expression to evaluate with values and operands seperated by a comma
Enter 'quit' to exit
10,6,9,3,+,-11,x,/,x,17,+,5,+
Stack contents:
10
Stack contents:
6
10
Stack contents:
9
6
10
Stack contents:
3
9
6
10
Stack contents:
12
6
10
Stack contents:
-11
12
6
10
Stack contents:
```

```
-132
6
10
Stack contents:
0
10
Stack contents:
0
Stack contents:
17
0
Stack contents:
17
Stack contents:
5
17
Stack contents:
22
Enter RPN expression to evaluate with values and operands seperated by a comma
Enter 'quit' to exit
quit

Process finished with exit code 0
```

Question 6

Time complexity:

	isprime()	Sieve of Eratosthenes
O(n)	O(n)	O(nlog(log n))

The `isprime` function checks whether a number is prime by checking if it is divisible by any number between 2 and half of the the number itself. The function `sieve_of_eratosthenes` implements the sieve of Eratosthenes algorithm. It iterates through all numbers from 2 to the square root of the number passed to it as a parameter and during each iteration it eliminates composite number (i.e. all numbers that multiples of the prime numbers). `Main` function contains the test driver.

Sieve of Eratosthenes adapted from: https://www.geeksforgeeks.org/sieve-of-eratosthenes/

Program:

```
import java.util.Arrays;

public class Question6 {
    public static void main(String[] args) {
        /* Test Driver */

        // testing isprime function
        System.out.println("Testing isprime() function...");
        for (int i = 2; i <= 20; i++) {
            if (isprime(i)) {
                System.out.println(i + " is prime");
            } else {
                System.out.println(i + " is not prime");
            }
        }
        System.out.println();

        // testing sieve of eratosthenes
        System.out.println("Testing sieve_of_eratosthenes() function...");
        boolean primes[] = sieve_of_eratosthenes(20);
        for (int i = 2; i < primes.length; i++) {
            if(prime[i]) {
                System.out.println(i + " is prime");
            } else {
                System.out.println(i + " is not prime");
            }
        }

    }

    // check if a number is prime
    static boolean isprime(int n) {
        // true if number is prime, false if not prime
        boolean prime = true;
            // loop through all numbers bigger than 1 and smaller than the
number divided by 2
            for (int i = 2; i <= n/2; i++) {
```

```
                // check if number is divisable by i
                if (n % i == 0) {
                    prime = false; // if remainder is 0, number isn't prime
                    break; // no need to continue looping
                }
            }
        return prime;
    }

    // Sieve of Eratosthenes algorithm
    // Adapted from: https://www.geeksforgeeks.org/sieve-of-eratosthenes/
    static boolean[] sieve_of_eratosthenes (int n) {
        boolean prime_numbers[] = new boolean[n+1]; // stores whether the
number at an index is prime or not
        Arrays.fill(prime_numbers, true); // initialise all numbers to true

        // set 0 and 1 as not being prime numbers
        prime_numbers[0] = false;
        prime_numbers[1] = false;

        // starting with 2 since it is the first prime number
        for (int i = 2; i*i <= n; i++) {
            if(prime_numbers[i]) {
                // mark all multiples of i as false since they are not
primes
                for(int j = i*i; j <= n; j = j + i) {
                    prime_numbers[j] = false;
                }
            }
        }

        return prime_numbers;
    }
}
```

Output:

```
/Library/Java/JavaVirtualMachines/temurin-17.jdk/Contents/Home/bin/java -javaagent:/Applications/IntelliJ IDEA.app/Contents/lib/idea_rt.jar=58619:/Applications/IntelliJ
 IDEA.app/Contents/bin -Dfile.encoding=UTF-8 -classpath /Users/lenisesilvio/IdeaProjects/ICT1018 Assignment/out/production/Question6 Question6
Testing isprime() function...
2 is prime
3 is prime
4 is not prime
5 is prime
6 is not prime
7 is prime
8 is not prime
9 is not prime
10 is not prime
11 is prime
12 is not prime
13 is prime
14 is not prime
15 is not prime
16 is not prime
17 is prime
18 is not prime
19 is prime
20 is not prime
```

```
Testing sieve_of_eratosthenes() function...
2 is prime
3 is prime
4 is not prime
5 is prime
6 is not prime
7 is prime
8 is not prime
9 is not prime
10 is not prime
11 is prime
12 is not prime
13 is prime
14 is not prime
15 is not prime
16 is not prime
17 is prime
18 is not prime
19 is prime
20 is not prime

Process finished with exit code 0
```

Question 7

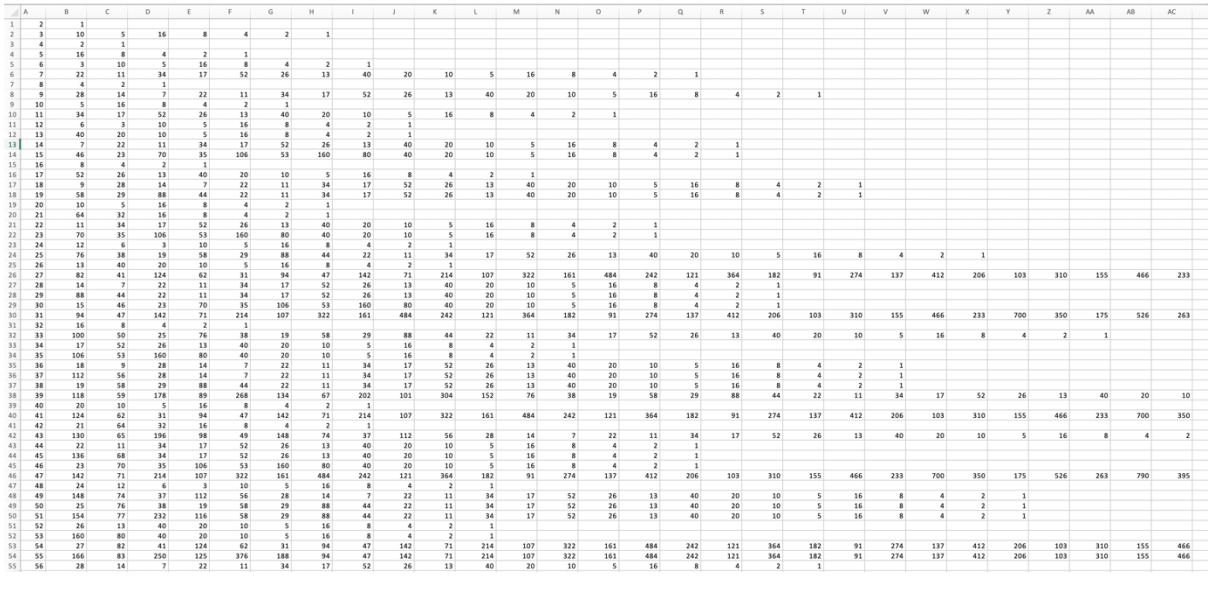
Time complexity: O(nlogn)

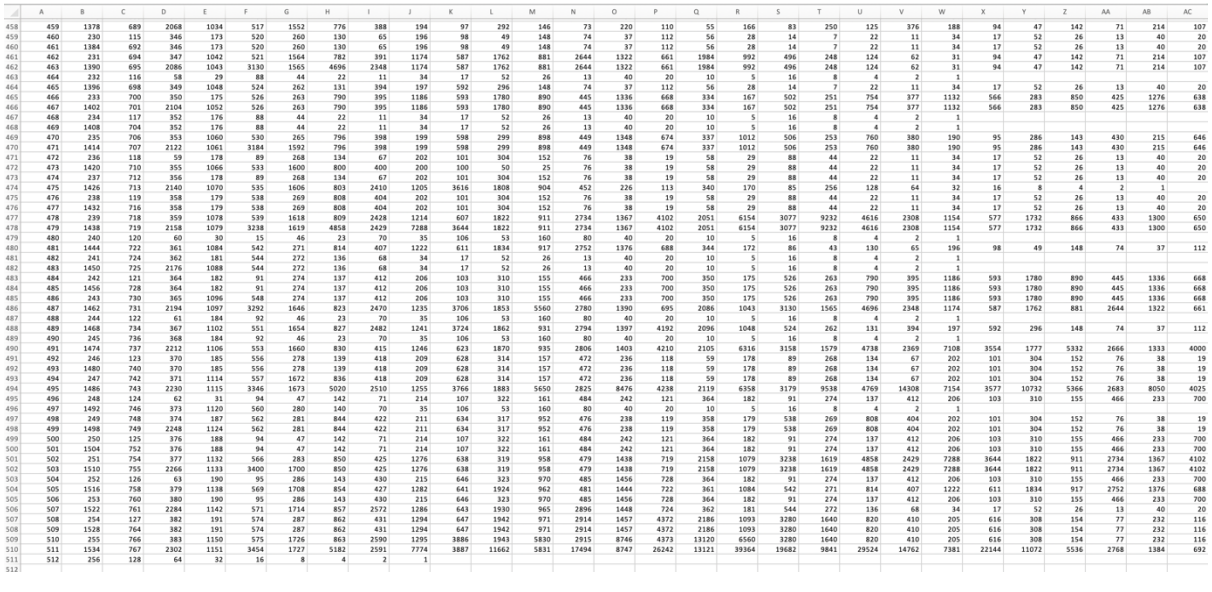
The program finds the Collatz sequence of every number between 2 and 512 and adds each sequence to a CSV file.

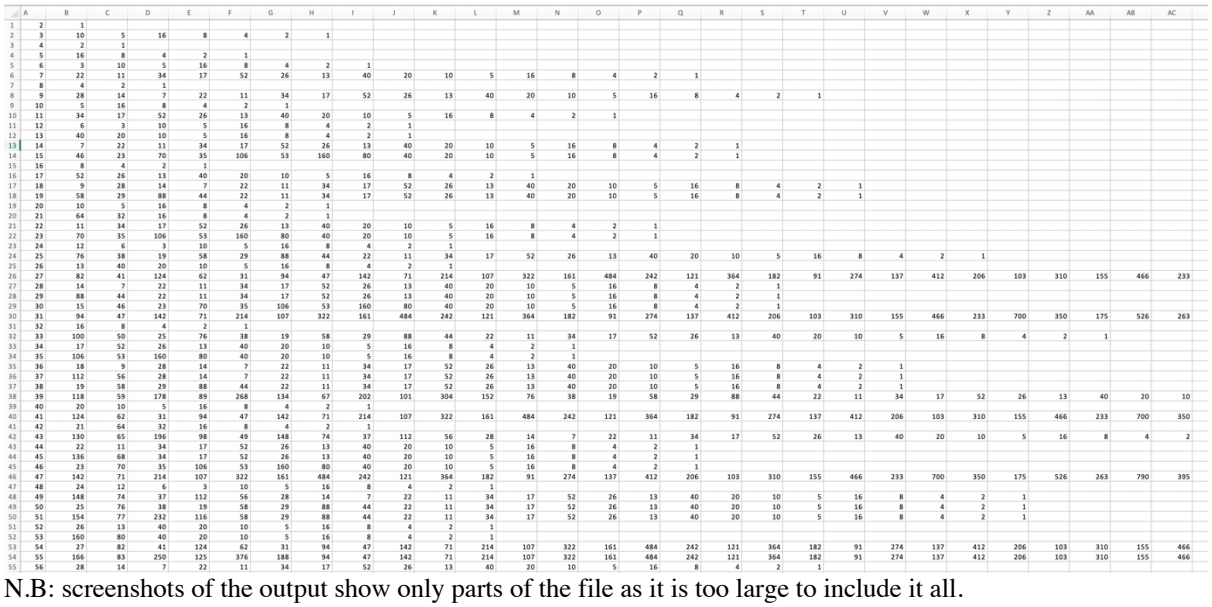
Program:

```
import java.io.File;
import java.io.FileWriter;
import java.io.IOException;

public class Question7 {
    public static void main(String[] args) throws IOException {
        // create and open file
        File collatzCSV = new File("Collatz.csv");

        FileWriter writer = new FileWriter(collatzCSV);

        // loop through all numbers between 2 and 512 and find their
collatz sequence
        // write sequence to CSV file
        for (int i = 2; i <= 512; i++) {

            int collatz_num = i;

            // add collatz sequence for i
            writer.append(Integer.toString(collatz_num));

            // keep repeating collatz rules until number is equal to 1
            // add number to csv file each time
            while (collatz_num != 1) {
                writer.append(','); // seperate values by a comma
                if (collatz_num % 2 == 0) { // if even
                    collatz_num = collatz_num/2;

                } else { // if odd
                    collatz_num = (3 * collatz_num) + 1;
                }
                writer.append(Integer.toString(collatz_num));
            }

            writer.append('\n');
            writer.flush();
        }
        writer.close(); //close file
    }
}
```

Output:







N.B: screenshots of the output show only parts of the file as it is too large to include it all.

Question 8

Time complexity: O(logn)

The function `square_root_approx` calculates the approximation of the square root using the Newton Raphson method. The `main` function contains driver used to test the function.

Adapted from: https://www.geeksforgeeks.org/find-root-of-a-number-using-newtons-method/

Program:

```
public class Question8 {
    public static void main(String[] args) {
        /* Test Driver */
        System.out.println("Square root of 90 is " +
square_root_approx(90));
        System.out.println("Square root of 4 is " + square_root_approx(4));
        System.out.println("Square root of 2 is " + square_root_approx(2));
        System.out.println("Square root of 100 is " +
square_root_approx(100));
        System.out.println("Square root of 27 is " +
square_root_approx(27));
        System.out.println("Square root of 198 is " +
square_root_approx(198));
    }

    // Approximates square root of number using Newton-Raphson method
    // Adapted from: https://www.geeksforgeeks.org/find-root-of-a-number-
using-newtons-method/
    static double square_root_approx(double number) {

        double root; // stores root
        double approx = number; // assumes approximation as being equal to
the number

        while (true) {
            root = 0.5 * (approx + (number/approx)); // recalculate root
approximation

            // If the difference between the current root and the previous
approximation is 0 stop
            if (Math.abs(root-approx) == 0) {
                break;
            }

            approx = root; // store current approximation for next
iteration
        }

        return root;
    }
}
```

Output:

```
/Library/Java/JavaVirtualMachines/temurin-17.jdk/Contents/Home/bin/java -javaagent:/Applications/IntelliJ IDEA.app/Contents/lib/idea_rt.jar=59923:/Applications/IntelliJ
 IDEA.app/Contents/bin -Dfile.encoding=UTF-8 -classpath /Users/lenisesilvio/IdeaProjects/ICT1018_Assignment/out/production/Question8 Question8
Square root of 90 is 9.486832980505138
Square root of 4 is 2.0
Square root of 2 is 1.414213562373095
Square root of 100 is 10.0
Square root of 27 is 5.196152422706632
Square root of 198 is 14.071247279470288

Process finished with exit code 0
```

Question 9

Time complexity: O(n)

The function `repeated_numbers` finds all repeated numbers by using a HashMap to keep track of whether a number is repeated or not. The function `print_repeated` prints whether a number is repeated or not to test the `repeated_numbers` function. The test driver is in `main`.

Program:

```
import java.util.HashMap;
public class Question9 {
    public static void main(String[] args) {
        /* Test Driver */
        int array1[] = {1, 2, 3, 4, 5, 6};
        System.out.println("Checking array1...");
        print_repeated(repeated_numbers(array1));
        System.out.println();

        int array2[] = {1, 1, 1, 1, 1, 1};
        System.out.println("Checking array2...");
        print_repeated(repeated_numbers(array2));
        System.out.println();

        int array3[] = {1, 2, 2, 4, 5, 6, 6, 9};
        System.out.println("Checking array3...");
        print_repeated(repeated_numbers(array3));
        System.out.println();

        int array4[] = {1, 2, 3, 1, 4, 5, 3, 6, 7, 8, 2};
        System.out.println("Checking array4...");
        print_repeated(repeated_numbers(array4));
        System.out.println();
    }

    // Check which numbers in an array are repeated or not
    static HashMap<Integer, Boolean> repeated_numbers(int array[]) {
        // stores numbers and if they are repeated or nopt
        HashMap<Integer, Boolean> repeated = new HashMap<>();

        // iterate through all numbers in the array
        for (int number: array) {
            // if not in hashmap
            if (repeated.get(number) == null) {
                repeated.put(number, false); // set number as not repeated
            } else { // if already in hashmap
                repeated.replace(number, false, true); // set number as
repeated
            }
        }
        return repeated;
    }

    // Prints whether number is repeated or not
    static void print_repeated (HashMap<Integer, Boolean> map) {
        for (int number: map.keySet()) { //iterate through all numbers in
```

```
hashmap and see if they are repeated or not
            if (map.get(number)) { // if repeated
                System.out.println(number + " is repeated");
            } else { // if not repeated
                System.out.println(number + " is not repeated");
            }
        }
    }
}
```

Output:

```
/Library/Java/JavaVirtualMachines/temurin-17.jdk/Contents/Home/bin/java -javaagent:/Applications/IntelliJ IDEA.app/Contents/lib/idea_rt.jar=59940:/Applications/IntelliJ IDEA.app/Contents/bin -Dfile.encoding=UTF-8 -classpath /Users/lenisesilvio/IdeaProjects/ICT1018 Assignment/out/production/Question9 Question9
Checking array1...
1 is not repeated
2 is not repeated
3 is not repeated
4 is not repeated
5 is not repeated
6 is not repeated

Checking array2...
1 is repeated

Checking array3...
1 is not repeated
2 is repeated
4 is not repeated
5 is not repeated
6 is repeated
9 is not repeated

Checking array4...
1 is repeated
2 is repeated
3 is repeated
4 is not repeated
5 is not repeated
6 is not repeated
7 is not repeated
8 is not repeated


Process finished with exit code 0
```

Question 10

Time complexity: O(n)

The function `get_largest` finds the largest number in an array recursively starting from the first element and comparing it to the return value of the next recursive function call. `main` contains the test driver.

Program:

```
public class Question10 {
    public static void main(String[] args) {
        /* Test driver */
        int array1[] = {1, 2, 3, 4, 5, 6};
        System.out.println("Largest number in array1 is " + get_largest(0,
array1));
        System.out.println();

        int array2[] = {1, 9, 3, 11, 100, 21, 5, 8};
        System.out.println("Largest number in array2 is " + get_largest(0,
array2));
        System.out.println();

        int array3[] = {9, 8, 7, 6, 5, 4};
        System.out.println("Largest number in array3 is " + get_largest(0,
array3));
        System.out.println();

        int array4[] = {11, 5, 7, 21, 15};
        System.out.println("Largest number in array4 is " + get_largest(0,
array4));
        System.out.println();

        int array5[] = {9, 3, 8, 6, 10, 4, 5};
        System.out.println("Largest number in array5 is " + get_largest(0,
array5));
    }

    // recursively find the largest number in an array
    static int get_largest(int index, int array[]) {

        // base case
        // if last two elements in the array are reached
        if (index == array.length-2) {
            // return the largest between them
            return Math.max(array[index], array[index+1]);
        }

        // recursive case
        // return the largest between the number and the return value of
the recursive call of the function
        return Math.max(get_largest(index+1, array), array[index]);
    }
}
```

Output:

```
/Library/Java/JavaVirtualMachines/temurin-17.jdk/Contents/Home/bin/java -javaagent:/Applications/IntelliJ IDEA.app/Contents/lib/idea_rt.jar=59958:/Applications/IntelliJ
 IDEA.app/Contents/bin -Dfile.encoding=UTF-8 -classpath /Users/lenisesilvio/IdeaProjects/ICT1018_Assignment/out/production/Question10 Question10
Largest number in array1 is 6

Largest number in array2 is 100

Largest number in array3 is 9

Largest number in array4 is 21

Largest number in array5 is 10

Process finished with exit code 0
```

Question 11

Time complexity: O(n)

The function `cosine_sine` calculates either the cosine or sine of an angle in radians using the summation of the Maclaurin series. The `main` contains the test driver used to test the function.

Program:

```
public class Question11 {
    public static void main(String[] args) {
        /* Test Driver */
        System.out.println("cos(-4) = " + cosine_sine(10, -4, 'c'));
        System.out.println("sin(6) = "+ cosine_sine(47, 6, 's'));
        System.out.println("cos(π/2) = " + cosine_sine(5, Math.PI/2, 'c'));
        System.out.println("sin(π/4) = "+ cosine_sine(6, Math.PI/4, 's'));
        System.out.println("cos(0.5) = " + cosine_sine(20, 0.5, 'c'));
        System.out.println("sin(-0.5) = "+ cosine_sine(36, -0.5, 's'));
    }

    // finds cosine or sine value using the maclaurin series
    static double cosine_sine (int r, double x, char t) {
        double cosine_sine = 0; // store cosine or sine value
        float factorial = 1; // store factorial

        // if calculating the cosine
        if (t == 'c') {
            cosine_sine = 1; // first number in series is 1
            for (int i = 1; i <= r; i++) { // find the specified number of
terms
                factorial *= (2*i) * (2*i-1); // calculate the factorial
                cosine_sine += Math.pow((-1), i) * (Math.pow(x,
(2*i))/factorial); // sum cosine mclaurin series terms
            }
        } else if (t == 's') { // if calculating the sine
            cosine_sine = x; // first number in series is equal to x
            for (int i = 1; i <= r; i++) { // find the specified number of
terms
                factorial *= (2*i) * ((2*i)+1); // calculate the factorial
                cosine_sine += Math.pow((-1), i) * (Math.pow(x,
(2*i)+1)/factorial); // sum sine mclaurin series terms
            }
        }
        return cosine_sine;
    }
}
```

Output:

```
/Library/Java/JavaVirtualMachines/temurin-17.jdk/Contents/Home/bin/java -javaagent:/Applications/IntelliJ IDEA.app/Contents/lib/idea_rt.jar=60002:/Applications/IntelliJ
 IDEA.app/Contents/bin -Dfile.encoding=UTF-8 -classpath /Users/lenisesilvio/IdeaProjects/ICT1018_Assignment/out/production/question11 Question11
cos(-4) = -0.6536436057137487
sin(6) = -0.2794154889133128
cos(π/2) = -4.6476600836607633E-7
sin(π/4) = 0.7071067811865679
cos(0.5) = 0.8775825618903728
sin(-0.5) = -0.479425538604203

Process finished with exit code 0
```

Question 12

Time complexity: O(n)

The function `fibonacci_sum` calculates the sum of the first n terms in the Fibonacci sequence. The function assumes the sequence starts from 1. The test driver is in `main`.

Program:

```
public class Question12 {
    public static void main(String[] args) {
        /* Test driver */
        System.out.println("The sum of the first 5 fibonacci numbers is: "
+ fibonacci_sum(5));
        System.out.println("The sum of the first 2 fibonacci numbers is: "
+ fibonacci_sum(2));
        System.out.println("The sum of the first 1 fibonacci numbers is: "
+ fibonacci_sum(1));
        System.out.println("The sum of the first 10 fibonacci numbers is: "
+ fibonacci_sum(10));
        System.out.println("The sum of the first 15 fibonacci numbers is: "
+ fibonacci_sum(15));
    }

    // Returns sum of fibonacci sequence
    static int fibonacci_sum(int n) {
        int sum = 0; // Stores sum of fibonacci sequence

        // fn_2 and fn_1 store the previous two numbers in the sequence
        int fn_2;
        int fn_1 = 0;

        int fn = 1; // store current number of the sequence

        // Calculates the sum until the last number in the sequence is
reached
        while (n > 1) {
            sum += fn; // add current number to sum
            fn_2 = fn_1; // move fn_1 to fn_2
            fn_1 = fn; // set current number as previous number
            fn = fn_2 + fn_1; // update current fibonacci number
            n--; // decrement counter
        }
        // Calculates last number and adds it to the sum
        sum += fn;

        return sum;
    }
}
```

Output:

```
/Library/Java/JavaVirtualMachines/temurin-17.jdk/Contents/Home/bin/java -javaagent:/Applications/IntelliJ IDEA.app/Contents/lib/idea_rt.jar=60019:/Applications/IntelliJ
 IDEA.app/Contents/bin -Dfile.encoding=UTF-8 -classpath /Users/lenisesilvio/IdeaProjects/ICT1018_Assignment/out/production/Question12 Question12
The sum of the first 5 fibonacci numbers is: 12
The sum of the first 2 fibonacci numbers is: 2
The sum of the first 1 fibonacci numbers is: 1
The sum of the first 10 fibonacci numbers is: 143
The sum of the first 15 fibonacci numbers is: 1596

Process finished with exit code 0
```