Lucrare de laborator NR.4

Scopul: Programarea algoritmilor de prelucrare a tablourilor aplicând tehnicile și metodele de sortare prin utilizarea funcțiilor, pointerilor, alocării dinamice a memoriei în limbajul C.

Sarcina(conform variantelor):
Elaborați un program C care va crea un meniu recursiv. Acesta trebuie să cuprindă următoarele funcții în C (cu apelare ulterioară ale acestora în funcția *main()*):

1. *Introducerea valorilor tabloului de la tastatură;*
2. *Completarea tabloului cu valori random;*
3. *Afișarea elementelor tabloului la ecran;*
4. *Sortarea elementelor tabloului conform variantelor:*
    1) Bubble Sort;
    2) Selection Sort;
    3) Insertion Sort;
    4) Merge Sort;
    5) Quick Sort;
    6) Shell Sort;
    7) Counting Sort;
    8) Heap Sort;
    9) Radix Sort;
    10) Comb Sort;
5. *Eliberarea memoriei și ieșirea din program.*

P.S. După fiecare manipulare a datelor să fie prevăzută afișarea rezultatelor ca și concluzie.
Algoritmul funcțiilor va fi programat, reieșind din conținutul problemei în două versiuni:

A. cu utilizarea metodei de transmitere a parametricelor funcțiilor prin valoare;
B. cu utilizarea metodei de transmitere a parametricelor funcțiilor prin adresă / Pointeri (parametrul formal va fi un pointer către valoarea obiectului corespunzător).

Variante spre rezolvare:
Varianta 1

1. Se dă un array unidimensional cu elemente de tip *integer* și un număr par *n* (condiția dată trebuie să se verifice în program!), valoarea căruia este citită de la tastatură. Să se afișeze array-ul original și array-ul modificat după fiecare manipulare a datelor din array, afișarea rezultatelor ca și concluzie (De ex.: Nu există numere prime; Nu există așa element cu indexul dat și alte date stipulate în condiția problemei) etc.

    A. Dacă media aritmetică a elementelor din prima jumătate a array-ului este mai mare decât madia aritmetică a elementelor din a doua jumătate a array-ului, atunci să se sorteze elementele array-ului crescător, aplicând tehnica de sortare HeapSort, altfel să se sorteze elementele array-ului descrescător, aplicând tehnica de sortare CountingSort.

B. Dacă suma elementelor pare a array-ului este mai mică decât suma elementelor impare din array, atunci să se sorteze elementele array-ului crescător, aplicând tehnica de sortare RadixSort, altfel să se sorteze elementele array-ului descrescător, aplicând tehnica de sortare CombSort.

C. Dacă există cel puțin două elemente numere prime, atunci să se sorteze elementele array-ului descrescător, aplicând tehnica de sortare MergeSort, altfel să se sorteze elementele array-ului crescător, aplicând tehnica de sortare BubbleSort.

2. Se dă un array bidimensional cu elemente de tip *integer* și un număr natural n ($n \geq 0$), valoarea căruia este citită de la tastatură. Să se afișeze array-ul original și array-ul modificat după fiecare manipulare a datelor din array, afișarea rezultatelor ca și concluzie (De ex.: Nu există numere prime; Nu există așa element cu indexul dat și alte date stipulate în condiția problemei) etc.

   A. Dacă numărul elementelor din array este mai mare decât valoarea lui k, declarată ca o constantă globală, atunci să se sorteze crescător elementele pe linie din array, aplicând tehnica de sortare QuickSort, altfel să se sorteze descrescător elementele pe coloane din array, aplicând tehnica de sortare ShellSort.

   B. Dacă elementul maxim din array se află cel puțin o dată pe linie pară, atunci să se sorteze crescător elementele liniei în care se află elementul maxim, aplicând tehnica de sortare SelectionSort, altfel să se sorteze descrescător elementele de pe coloana a doua, aplicând tehnica de sortare InsertionSort.

Varianta 2

1. Se dă un array unidimensional cu elemente de tip *integer* și un număr impar n (condiția dată trebuie să se verifice în program!), valoarea căruia este citită de la tastatură. Să se afișeze array-ul original și array-ul modificat după fiecare manipulare a datelor din array, afișarea rezultatelor ca și concluzie (De ex.: Nu există numere prime; Nu există așa element cu indexul dat și alte date stipulate în condiția problemei) etc.

   A. Dacă primul element, număr negativ, întâlnit în array, se află pe poziție pară, atunci să se sorteze elementele array-ului crescător, aplicând tehnica de sortare HeapSort, altfel să se sorteze elementele array-ului descrescător, aplicând tehnica de sortare CountingSort.

   B. Dacă suma elementelor pare a array-ului este mai mică decât media aritmetică a elementelor impare din array, atunci să se sorteze elementele array-ului descrescător, aplicând tehnica de sortare RadixSort, altfel să se sorteze elementele array-ului crescător, aplicând tehnica de sortare CombSort.

   C. Dacă există cel puțin două elemente numere prime, atunci să se sorteze elementele array-ului crescător, aplicând tehnica de sortare MergeSort, altfel să se sorteze elementele array-ului descrescător, aplicând tehnica de sortare BubbleSort.

2. Se dă un array bidimensional cu elemente de tip *integer* și un număr natural n ($n \geq 0$), valoarea căruia este citită de la tastatură. Să se afișeze array-ul original și array-ul modificat după fiecare manipulare a datelor din array, afișarea rezultatelor ca și

concluzie (De ex.: Nu există numere prime; Nu există așa element cu indexul dat și alte date stipulate în condiția problemei) etc.

A. Dacă produsul elementelor de pe prima coloană este mai mare decât valoarea lui *k*, declarată ca o constantă globală, atunci să se sorteze crescător elementele de pe diagonala principală a array-ului, aplicând tehnica de sortare QuickSort, altfel să se sorteze descrescător elementele de pe diagonala secundară a array-ului, aplicând tehnica de sortare ShellSort.

B. Dacă elementul minim din array se află cel puțin o dată pe linie impară, atunci să se sorteze crescător elementele liniei în care se află elementul minim, aplicând tehnica de sortare SelectionSort, altfel să se sorteze descrescător elementele de pe coloana în care se află elementul minim, aplicând tehnica de sortare InsertionSort.

Varianta 3

1. Se dă un array unidimensional cu elemente de tip *integer* și un număr par *n* (condiția dată trebuie să se verifice în program!), valoarea căruia este citită de la tastatură. Să se afișeze array-ul original și array-ul modificat după fiecare manipulare a datelor din array, afișarea rezultatelor ca și concluzie (De ex.: Nu există numere prime; Nu există așa element cu indexul dat și alte date stipulate în condiția problemei) etc.

   A. Dacă există cel puțin un element nul, atunci să se sorteze elementele array-ului descrescător, aplicând tehnica de sortare HeapSort, altfel să se sorteze elementele array-ului crescător, aplicând tehnica de sortare CountingSort.

   B. Dacă produsul elementelor impare a array-ului este un număr pozitiv (array-ul va conține numere atât pozitive, cât și negative), atunci să se sorteze elementele array-ului crescător, aplicând tehnica de sortare RadixSort, altfel să se sorteze elementele array-ului descrescător, aplicând tehnica de sortare CombSort.

   C. Dacă există cel puțin un element număr prim, atunci să se sorteze elementele array-ului descrescător, aplicând tehnica de sortare MergeSort, altfel să se sorteze elementele array-ului crescător, aplicând tehnica de sortare BubbleSort.

2. Se dă un array bidimensional cu elemente de tip *integer* și un număr natural *n* ($n \geq 0$), valoarea căruia este citită de la tastatură. Să se afișeze array-ul original și array-ul modificat după fiecare manipulare a datelor din array, afișarea rezultatelor ca și concluzie (De ex.: Nu există numere prime; Nu există așa element cu indexul dat și alte date stipulate în condiția problemei) etc.

   A. Dacă produsul elementelor de sub diagonala principală este mai mare decât valoarea lui *k*, declarată ca o constantă globală, atunci să se sorteze crescător elementele de pe diagonala principală a array-ului, aplicând tehnica de sortare QuickSort, altfel să se sorteze descrescător elementele de pe diagonala secundară a array-ului, aplicând tehnica de sortare ShellSort.

   B. Dacă elementul minim din array se întâlnește numai o singură dată în array, atunci să se sorteze crescător elementele liniei în care se află elementul minim, aplicând tehnica de sortare SelectionSort, altfel să se sorteze descrescător elementele de pe coloanele în care se află elementul minim, aplicând tehnica de sortare InsertionSort.

Varianta 4

1. Se dă un array unidimensional cu elemente de tip *integer* și un număr natural *n*, valoarea căruia este citită de la tastatură. Să se afișeze array-ul original și array-ul modificat după fiecare manipulare a datelor din array, afișarea rezultatelor ca și concluzie (De ex.: Nu există numere prime; Nu există așa element cu indexul dat și alte date stipulate în condiția problemei) etc.

   A. Dacă media aritmetică a elementelor de pe poziții pare este mai mare decât media aritmetică a elementelor de pe pozițiile impare, atunci să se sorteze elementele array-ului crescător, aplicând tehnica de sortare HeapSort, altfel să se sorteze elementele array-ului descrescător, aplicând tehnica de sortare CountingSort.

   B. Dacă există numere prime în array, atunci să se sorteze elementele array-ului crescător, aplicând tehnica de sortare RadixSort, altfel să se sorteze elementele array-ului descrescător, aplicând tehnica de sortare CombSort.

   C. Dacă produsul elementelor negative este un număr negativ (array va conține atât elemente pozitive, cât și elemente negative), atunci să se sorteze elementele array-ului descrescător, aplicând tehnica de sortare MergeSort, altfel să se sorteze elementele array-ului crescător, aplicând tehnica de sortare BubbleSort.

2. Se dă un array bidimensional cu elemente de tip *integer* și un număr natural n ($n \geq 0$), valoarea căruia este citită de la tastatură. Să se afișeze array-ul original și array-ul modificat după fiecare manipulare a datelor din array, afișarea rezultatelor ca și concluzie (De ex.: Nu există numere prime; Nu există așa element cu indexul dat și alte date stipulate în condiția problemei) etc.

   A. Dacă numărul elementelor, aflate deasupra diagonalei principale, este mai mare decât valoarea lui *k*, declarată ca o constantă globală, atunci să se sorteze crescător elementele de pe diagonala secundară, aplicând tehnica de sortare QuickSort, altfel să se sorteze descrescător elementele din prima coloană, aplicând tehnica de sortare ShellSort.

   B. Dacă elementul maxim se găsește o singură dată în array, atunci să se sorteze crescător elementele liniei în care se află elementul maxim, aplicând tehnica de sortare SelectionSort, altfel să se sorteze descrescător elementele de pe coloanele în care se află elementul maxim, aplicând tehnica de sortare InsertionSort.

Varianta 5

1. Se dă un array unidimensional cu elemente de tip *integer* și un număr par *n* (condiția dată trebuie să se verifice în program!), valoarea căruia este citită de la tastatură. Să se afișeze array-ul original și array-ul modificat după fiecare manipulare a datelor din array, afișarea rezultatelor ca și concluzie (De ex.: Nu există numere prime; Nu există așa element cu indexul dat și alte date stipulate în condiția problemei) etc.

   A. Dacă primul element, număr negativ, întâlnit în array, se află pe poziție pară, atunci să se sorteze elementele array-ului crescător, aplicând tehnica de sortare HeapSort, altfel să se sorteze elementele array-ului descrescător, aplicând tehnica de sortare CountingSort.

B. Dacă suma elementelor pare a array-ului este mai mică decât media aritmetică a elementelor impare din array, atunci să se sorteze elementele array-ului descrescător, aplicând tehnica de sortare RadixSort, altfel să se sorteze elementele array-ului crescător, aplicând tehnica de sortare CombSort.

C. Dacă există cel puțin două elemente numere prime, atunci să se sorteze elementele array-ului crescător, aplicând tehnica de sortare MergeSort, altfel să se sorteze elementele array-ului descrescător, aplicând tehnica de sortare BubbleSort.

2. Se dă un array bidimensional cu elemente de tip *integer* și un număr natural n ($n \geq 0$), valoarea căruia este citită de la tastatură. Să se afișeze array-ul original și array-ul modificat după fiecare manipulare a datelor din array, afișarea rezultatelor ca și concluzie (De ex.: Nu există numere prime; Nu există așa element cu indexul dat și alte date stipulate în condiția problemei) etc.

A. Dacă produsul elementelor de pe prima coloană este mai mare decât valoarea lui k, declarată ca o constantă globală, atunci să se sorteze crescător elementele de pe diagonala principală a array-ului, aplicând tehnica de sortare QuickSort, altfel să se sorteze descrescător elementele de pe diagonala secundară a array-ului, aplicând tehnica de sortare ShellSort.

B. Dacă elementul minim din array se află cel puțin o dată pe linie impară, atunci să se sorteze crescător elementele liniei în care se află elementul minim, aplicând tehnica de sortare SelectionSort, altfel să se sorteze descrescător elementele de pe coloana în care se află elementul minim, aplicând tehnica de sortare InsertionSort.

Varianta 6

1. Se dă un array unidimensional cu elemente de tip *integer* și un număr impar n (condiția dată trebuie să se verifice în program!), valoarea căruia este citită de la tastatură. Să se afișeze array-ul original și array-ul modificat după fiecare manipulare a datelor din array, afișarea rezultatelor ca și concluzie (De ex.: Nu există numere prime; Nu există așa element cu indexul dat și alte date stipulate în condiția problemei) etc.

A. Dacă există cel puțin un element nul, atunci să se sorteze elementele array-ului descrescător, aplicând tehnica de sortare HeapSort, altfel să se sorteze elementele array-ului crescător, aplicând tehnica de sortare CountingSort.

B. Dacă produsul elementelor impare a array-ului este un număr pozitiv (array-ul va conține numere atât pozitive, cât și negative), atunci să se sorteze elementele array-ului crescător, aplicând tehnica de sortare RadixSort, altfel să se sorteze elementele array-ului descrescător, aplicând tehnica de sortare CombSort.

C. Dacă există cel puțin un element număr prim, atunci să se sorteze elementele array-ului descrescător, aplicând tehnica de sortare MergeSort, altfel să se sorteze elementele array-ului crescător, aplicând tehnica de sortare BubbleSort.

2. Se dă un array bidimensional cu elemente de tip *integer* și un număr natural n ($n \geq 0$), valoarea căruia este citită de la tastatură. Să se afișeze array-ul original și array-ul modificat după fiecare manipulare a datelor din array, afișarea rezultatelor ca și

concluzie (De ex.: Nu există numere prime; Nu există așa element cu indexul dat și alte date stipulate în condiția problemei) etc.

A. Dacă produsul elementelor de sub diagonala principală este mai mare decât valoarea lui *k*, declarată ca o constantă globală, atunci să se sorteze crescător elementele de pe diagonala principală a array-ului, aplicând tehnica de sortare QuickSort, altfel să se sorteze descrescător elementele de pe diagonala secundară a array-ului, aplicând tehnica de sortare ShellSort.

B. Dacă elementul minim din array se întâlnește numai o singură dată în array, atunci să se sorteze crescător elementele liniei în care se află elementul minim, aplicând tehnica de sortare SelectionSort, altfel să se sorteze descrescător elementele de pe coloanele în care se află elementul minim, aplicând tehnica de sortare InsertionSort.

Varianta 7

1. Se dă un array unidimensional cu elemente de tip *integer* și un număr par *n* (condiția dată trebuie să se verifice în program!), valoarea căruia este citită de la tastatură. Să se afișeze array-ul original și array-ul modificat după fiecare manipulare a datelor din array, afișarea rezultatelor ca și concluzie (De ex.: Nu există numere prime; Nu există așa element cu indexul dat și alte date stipulate în condiția problemei) etc.

   A. Dacă media aritmetică a elementelor din prima jumătate a array-ului este mai mare decât madia aritmetică a elementelor din a doua jumătate a array-ului, atunci să se sorteze elementele array-ului crescător, aplicând tehnica de sortare HeapSort, altfel să se sorteze elementele array-ului descrescător, aplicând tehnica de sortare CountingSort.

   B. Dacă suma elementelor pare a array-ului este mai mică decât suma elementelor impare din array, atunci să se sorteze elementele array-ului crescător, aplicând tehnica de sortare RadixSort, altfel să se sorteze elementele array-ului descrescător, aplicând tehnica de sortare CombSort.

   C. Dacă există cel puțin două elemente numere prime, atunci să se sorteze elementele array-ului descrescător, aplicând tehnica de sortare MergeSort, altfel să se sorteze elementele array-ului crescător, aplicând tehnica de sortare BubbleSort.

2. Se dă un array bidimensional cu elemente de tip *integer* și un număr natural n ($n \geq 0$), valoarea căruia este citită de la tastatură. Să se afișeze array-ul original și array-ul modificat după fiecare manipulare a datelor din array, afișarea rezultatelor ca și concluzie (De ex.: Nu există numere prime; Nu există așa element cu indexul dat și alte date stipulate în condiția problemei) etc.

   A. Dacă numărul elementelor din array este mai mare decât valoarea lui *k*, declarată ca o constantă globală, atunci să se sorteze crescător elementele pe linie din array, aplicând tehnica de sortare QuickSort, altfel să se sorteze descrescător elementele pe coloane din array, aplicând tehnica de sortare ShellSort.

   B. Dacă elementul maxim din array se află cel puțin o dată pe linie pară, atunci să se sorteze crescător elementele liniei în care se află elementul maxim, aplicând

tehnica de sortare SelectionSort, altfel să se sorteze descrescător elementele de pe coloana a doua, aplicând tehnica de sortare InsertionSort.

Varianta 8

1. Se dă un array unidimensional cu elemente de tip *integer* și un număr impar *n* (condiția dată trebuie să se verifice în program!), valoarea căruia este citită de la tastatură. Să se afișeze array-ul original și array-ul modificat după fiecare manipulare a datelor din array, afișarea rezultatelor ca și concluzie (De ex.: Nu există numere prime; Nu există așa element cu indexul dat și alte date stipulate în condiția problemei) etc.

   A. Dacă primul element, număr negativ, întâlnit în array, se află pe poziție pară, atunci să se sorteze elementele array-ului crescător, aplicând tehnica de sortare HeapSort, altfel să se sorteze elementele array-ului descrescător, aplicând tehnica de sortare CountingSort.

   B. Dacă suma elementelor pare a array-ului este mai mică decât media aritmetică a elementelor impare din array, atunci să se sorteze elementele array-ului descrescător, aplicând tehnica de sortare RadixSort, altfel să se sorteze elementele array-ului crescător, aplicând tehnica de sortare CombSort.

   C. Dacă există cel puțin două elemente numere prime, atunci să se sorteze elementele array-ului crescător, aplicând tehnica de sortare MergeSort, altfel să se sorteze elementele array-ului descrescător, aplicând tehnica de sortare BubbleSort.

2. Se dă un array bidimensional cu elemente de tip *integer* și un număr natural *n* ($n \geq 0$), valoarea căruia este citită de la tastatură. Să se afișeze array-ul original și array-ul modificat după fiecare manipulare a datelor din array, afișarea rezultatelor ca și concluzie (De ex.: Nu există numere prime; Nu există așa element cu indexul dat și alte date stipulate în condiția problemei) etc.

   A. Dacă produsul elementelor de pe prima coloană este mai mare decât valoarea lui *k*, declarată ca o constantă globală, atunci să se sorteze crescător elementele de pe diagonala principală a array-ului, aplicând tehnica de sortare QuickSort, altfel să se sorteze descrescător elementele de pe diagonala secundară a array-ului, aplicând tehnica de sortare ShellSort.

   B. Dacă elementul minim din array se află cel puțin o dată pe linie impară, atunci să se sorteze crescător elementele liniei în care se află elementul minim, aplicând tehnica de sortare SelectionSort, altfel să se sorteze descrescător elementele de pe coloana în care se află elementul minim, aplicând tehnica de sortare InsertionSort.

Varianta 9

1. Se dă un array unidimensional cu elemente de tip *integer* și un număr par *n* (condiția dată trebuie să se verifice în program!), valoarea căruia este citită de la tastatură. Să se afișeze array-ul original și array-ul modificat după fiecare manipulare a datelor din array, afișarea rezultatelor ca și concluzie (De ex.: Nu există numere prime; Nu există așa element cu indexul dat și alte date stipulate în condiția problemei) etc.

A. Dacă există cel puțin un element nul, atunci să se sorteze elementele array-ului descrescător, aplicând tehnica de sortare HeapSort, altfel să se sorteze elementele array-ului crescător, aplicând tehnica de sortare CountingSort.

B. Dacă produsul elementelor impare a array-ului este un număr pozitiv (array-ul va conține numere atât pozitive, cât și negative), atunci să se sorteze elementele array-ului crescător, aplicând tehnica de sortare RadixSort, altfel să se sorteze elementele array-ului descrescător, aplicând tehnica de sortare CombSort.

C. Dacă există cel puțin un element număr prim, atunci să se sorteze elementele array-ului descrescător, aplicând tehnica de sortare MergeSort, altfel să se sorteze elementele array-ului crescător, aplicând tehnica de sortare BubbleSort.

2. Se dă un array bidimensional cu elemente de tip *integer* și un număr natural n ($n \geq 0$), valoarea căruia este citită de la tastatură. Să se afișeze array-ul original și array-ul modificat după fiecare manipulare a datelor din array, afișarea rezultatelor ca și concluzie (De ex.: Nu există numere prime; Nu există așa element cu indexul dat și alte date stipulate în condiția problemei) etc.

A. Dacă produsul elementelor de sub diagonala principală este mai mare decât valoarea lui k, declarată ca o constantă globală, atunci să se sorteze crescător elementele de pe diagonala principală a array-ului, aplicând tehnica de sortare QuickSort, altfel să se sorteze descrescător elementele de pe diagonala secundară a array-ului, aplicând tehnica de sortare ShellSort.

B. Dacă elementul minim din array se întâlnește numai o singură dată în array, atunci să se sorteze crescător elementele liniei în care se află elementul minim, aplicând tehnica de sortare SelectionSort, altfel să se sorteze descrescător elementele de pe coloanele în care se află elementul minim, aplicând tehnica de sortare InsertionSort.

Varianta 10

1. Se dă un array unidimensional cu elemente de tip *integer* și un număr natural n, valoarea căruia este citită de la tastatură. Să se afișeze array-ul original și array-ul modificat după fiecare manipulare a datelor din array, afișarea rezultatelor ca și concluzie (De ex.: Nu există numere prime; Nu există așa element cu indexul dat și alte date stipulate în condiția problemei) etc.

A. Dacă media aritmetică a elementelor de pe poziții pare este mai mare decât media aritmetică a elementelor de pe pozițiile impare, atunci să se sorteze elementele array-ului crescător, aplicând tehnica de sortare HeapSort, altfel să se sorteze elementele array-ului descrescător, aplicând tehnica de sortare CountingSort.

B. Dacă există numere prime în array, atunci să se sorteze elementele array-ului crescător, aplicând tehnica de sortare RadixSort, altfel să se sorteze elementele array-ului descrescător, aplicând tehnica de sortare CombSort.

C. Dacă produsul elementelor negative este un număr negativ (array va conține atât elemente pozitive, cât și elemente negative), atunci să se sorteze elementele array-ului descrescător, aplicând tehnica de sortare MergeSort, altfel să se sorteze elementele array-ului crescător, aplicând tehnica de sortare BubbleSort.

2. Se dă un array bidimensional cu elemente de tip *integer* și un număr natural n ($n \geq 0$), valoarea căruia este citită de la tastatură. Să se afișeze array-ul original și array-ul modificat după fiecare manipulare a datelor din array, afișarea rezultatelor ca și concluzie (De ex.: Nu există numere prime; Nu există așa element cu indexul dat și alte date stipulate în condiția problemei) etc.

   A. Dacă numărul elementelor, aflate deasupra diagonalei principale, este mai mare decât valoarea lui k, declarată ca o constantă globală, atunci să se sorteze crescător elementele de pe diagonala secundară, aplicând tehnica de sortare QuickSort, altfel să se sorteze descrescător elementele din prima coloană, aplicând tehnica de sortare ShellSort.

   B. Dacă elementul maxim se găsește o singură dată în array, atunci să se sorteze crescător elementele liniei în care se află elementul maxim, aplicând tehnica de sortare SelectionSort, altfel să se sorteze descrescător elementele de pe coloanele în care se află elementul maxim, aplicând tehnica de sortare InsertionSort.

Varianta 11

1. Se dă un array unidimensional cu elemente de tip *integer* și un număr par n (condiția dată trebuie să se verifice în program!), valoarea căruia este citită de la tastatură. Să se afișeze array-ul original și array-ul modificat după fiecare manipulare a datelor din array, afișarea rezultatelor ca și concluzie (De ex.: Nu există numere prime; Nu există așa element cu indexul dat și alte date stipulate în condiția problemei) etc.

   A. Dacă media aritmetică a elementelor din prima jumătate a array-ului este mai mare decât madia aritmetică a elementelor din a doua jumătate a array-ului, atunci să se sorteze elementele array-ului crescător, aplicând tehnica de sortare HeapSort, altfel să se sorteze elementele array-ului descrescător, aplicând tehnica de sortare CountingSort.

   B. Dacă suma elementelor pare a array-ului este mai mică decât suma elementelor impare din array, atunci să se sorteze elementele array-ului crescător, aplicând tehnica de sortare RadixSort, altfel să se sorteze elementele array-ului descrescător, aplicând tehnica de sortare CombSort.

   C. Dacă există cel puțin două elemente numere prime, atunci să se sorteze elementele array-ului descrescător, aplicând tehnica de sortare MergeSort, altfel să se sorteze elementele array-ului crescător, aplicând tehnica de sortare BubbleSort.

2. Se dă un array bidimensional cu elemente de tip *integer* și un număr natural n ($n \geq 0$), valoarea căruia este citită de la tastatură. Să se afișeze array-ul original și array-ul modificat după fiecare manipulare a datelor din array, afișarea rezultatelor ca și concluzie (De ex.: Nu există numere prime; Nu există așa element cu indexul dat și alte date stipulate în condiția problemei) etc.

   A. Dacă numărul elementelor din array este mai mare decât valoarea lui k, declarată ca o constantă globală, atunci să se sorteze crescător elementele pe linie din array, aplicând tehnica de sortare QuickSort, altfel să se sorteze descrescător elementele pe coloane din array, aplicând tehnica de sortare ShellSort.

B. Dacă elementul maxim din array se află cel puțin o dată pe linie pară, atunci să se sorteze crescător elementele liniei în care se află elementul maxim, aplicând tehnica de sortare SelectionSort, altfel să se sorteze descrescător elementele de pe coloana a doua, aplicând tehnica de sortare InsertionSort.

Varianta 12

1. Se dă un array unidimensional cu elemente de tip *integer* și un număr impar *n* (condiția dată trebuie să se verifice în program!), valoarea căruia este citită de la tastatură. Să se afișeze array-ul original și array-ul modificat după fiecare manipulare a datelor din array, afișarea rezultatelor ca și concluzie (De ex.: Nu există numere prime; Nu există așa element cu indexul dat și alte date stipulate în condiția problemei) etc.

   A. Dacă primul element, număr negativ, întâlnit în array, se află pe poziție pară, atunci să se sorteze elementele array-ului crescător, aplicând tehnica de sortare HeapSort, altfel să se sorteze elementele array-ului descrescător, aplicând tehnica de sortare CountingSort.

   B. Dacă suma elementelor pare a array-ului este mai mică decât media aritmetică a elementelor impare din array, atunci să se sorteze elementele array-ului descrescător, aplicând tehnica de sortare RadixSort, altfel să se sorteze elementele array-ului crescător, aplicând tehnica de sortare CombSort.

   C. Dacă există cel puțin două elemente numere prime, atunci să se sorteze elementele array-ului crescător, aplicând tehnica de sortare MergeSort, altfel să se sorteze elementele array-ului descrescător, aplicând tehnica de sortare BubbleSort.

2. Se dă un array bidimensional cu elemente de tip *integer* și un număr natural *n* ($n \geq 0$), valoarea căruia este citită de la tastatură. Să se afișeze array-ul original și array-ul modificat după fiecare manipulare a datelor din array, afișarea rezultatelor ca și concluzie (De ex.: Nu există numere prime; Nu există așa element cu indexul dat și alte date stipulate în condiția problemei) etc.

   A. Dacă produsul elementelor de pe prima coloană este mai mare decât valoarea lui *k*, declarată ca o constantă globală, atunci să se sorteze crescător elementele de pe diagonala principală a array-ului, aplicând tehnica de sortare QuickSort, altfel să se sorteze descrescător elementele de pe diagonala secundară a array-ului, aplicând tehnica de sortare ShellSort.

   B. Dacă elementul minim din array se află cel puțin o dată pe linie impară, atunci să se sorteze crescător elementele liniei în care se află elementul minim, aplicând tehnica de sortare SelectionSort, altfel să se sorteze descrescător elementele de pe coloana în care se află elementul minim, aplicând tehnica de sortare InsertionSort.

Varianta 13

1. Se dă un array unidimensional cu elemente de tip *integer* și un număr par *n* (condiția dată trebuie să se verifice în program!), valoarea căruia este citită de la tastatură. Să se afișeze array-ul original și array-ul modificat după fiecare manipulare a datelor din

array, afișarea rezultatelor ca și concluzie (De ex.: Nu există numere prime; Nu există așa element cu indexul dat și alte date stipulate în condiția problemei) etc.

A. Dacă există cel puțin un element nul, atunci să se sorteze elementele array-ului descrescător, aplicând tehnica de sortare HeapSort, altfel să se sorteze elementele array-ului crescător, aplicând tehnica de sortare CountingSort.

B. Dacă produsul elementelor impare a array-ului este un număr pozitiv (array-ul va conține numere atât pozitive, cât și negative), atunci să se sorteze elementele array-ului crescător, aplicând tehnica de sortare RadixSort, altfel să se sorteze elementele array-ului descrescător, aplicând tehnica de sortare CombSort.

C. Dacă există cel puțin un element număr prim, atunci să se sorteze elementele array-ului descrescător, aplicând tehnica de sortare MergeSort, altfel să se sorteze elementele array-ului crescător, aplicând tehnica de sortare BubbleSort.

2. Se dă un array bidimensional cu elemente de tip *integer* și un număr natural n ($n \geq 0$), valoarea căruia este citită de la tastatură. Să se afișeze array-ul original și array-ul modificat după fiecare manipulare a datelor din array, afișarea rezultatelor ca și concluzie (De ex.: Nu există numere prime; Nu există așa element cu indexul dat și alte date stipulate în condiția problemei) etc.

A. Dacă produsul elementelor de sub diagonala principală este mai mare decât valoarea lui *k*, declarată ca o constantă globală, atunci să se sorteze crescător elementele de pe diagonala principală a array-ului, aplicând tehnica de sortare QuickSort, altfel să se sorteze descrescător elementele de pe diagonala secundară a array-ului, aplicând tehnica de sortare ShellSort.

B. Dacă elementul minim din array se întâlnește numai o singură dată în array, atunci să se sorteze crescător elementele liniei în care se află elementul minim, aplicând tehnica de sortare SelectionSort, altfel să se sorteze descrescător elementele de pe coloanele în care se află elementul minim, aplicând tehnica de sortare InsertionSort.

Varianta 14

1. Se dă un array unidimensional cu elemente de tip *integer* și un număr natural *n*, valoarea căruia este citită de la tastatură. Să se afișeze array-ul original și array-ul modificat după fiecare manipulare a datelor din array, afișarea rezultatelor ca și concluzie (De ex.: Nu există numere prime; Nu există așa element cu indexul dat și alte date stipulate în condiția problemei) etc.

A. Dacă media aritmetică a elementelor de pe poziții pare este mai mare decât media aritmetică a elementelor de pe pozițiile impare, atunci să se sorteze elementele array-ului crescător, aplicând tehnica de sortare HeapSort, altfel să se sorteze elementele array-ului descrescător, aplicând tehnica de sortare CountingSort.

B. Dacă există numere prime în array, atunci să se sorteze elementele array-ului crescător, aplicând tehnica de sortare RadixSort, altfel să se sorteze elementele array-ului descrescător, aplicând tehnica de sortare CombSort.

C. Dacă produsul elementelor negative este un număr negativ (array va conține atât elemente pozitive, cât și elemente negative), atunci să se sorteze elementele array-

ului descrescător, aplicând tehnica de sortare MergeSort, altfel să se sorteze elementele array-ului crescător, aplicând tehnica de sortare BubbleSort.

2. Se dă un array bidimensional cu elemente de tip *integer* și un număr natural *n* ($n \geq 0$), valoarea căruia este citită de la tastatură. Să se afișeze array-ul original și array-ul modificat după fiecare manipulare a datelor din array, afișarea rezultatelor ca și concluzie (De ex.: Nu există numere prime; Nu există așa element cu indexul dat și alte date stipulate în condiția problemei) etc.

   A. Dacă numărul elementelor, aflate deasupra diagonalei principale, este mai mare decât valoarea lui *k*, declarată ca o constantă globală, atunci să se sorteze crescător elementele de pe diagonala secundară, aplicând tehnica de sortare QuickSort, altfel să se sorteze descrescător elementele din prima coloană, aplicând tehnica de sortare ShellSort.

   B. Dacă elementul maxim se găsește o singură dată în array, atunci să se sorteze crescător elementele liniei în care se află elementul maxim, aplicând tehnica de sortare SelectionSort, altfel să se sorteze descrescător elementele de pe coloanele în care se află elementul maxim, aplicând tehnica de sortare InsertionSort.

Varianta 15

1. Se dă un array unidimensional cu elemente de tip *integer* și un număr par *n* (condiția dată trebuie să se verifice în program!), valoarea căruia este citită de la tastatură. Să se afișeze array-ul original și array-ul modificat după fiecare manipulare a datelor din array, afișarea rezultatelor ca și concluzie (De ex.: Nu există numere prime; Nu există așa element cu indexul dat și alte date stipulate în condiția problemei) etc.

   A. Dacă primul element, număr negativ, întâlnit în array, se află pe poziție pară, atunci să se sorteze elementele array-ului crescător, aplicând tehnica de sortare HeapSort, altfel să se sorteze elementele array-ului descrescător, aplicând tehnica de sortare CountingSort.

   B. Dacă suma elementelor pare a array-ului este mai mică decât media aritmetică a elementelor impare din array, atunci să se sorteze elementele array-ului descrescător, aplicând tehnica de sortare RadixSort, altfel să se sorteze elementele array-ului crescător, aplicând tehnica de sortare CombSort.

   C. Dacă există cel puțin două elemente numere prime, atunci să se sorteze elementele array-ului crescător, aplicând tehnica de sortare MergeSort, altfel să se sorteze elementele array-ului descrescător, aplicând tehnica de sortare BubbleSort.

2. Se dă un array bidimensional cu elemente de tip *integer* și un număr natural *n* ($n \geq 0$), valoarea căruia este citită de la tastatură. Să se afișeze array-ul original și array-ul modificat după fiecare manipulare a datelor din array, afișarea rezultatelor ca și concluzie (De ex.: Nu există numere prime; Nu există așa element cu indexul dat și alte date stipulate în condiția problemei) etc.

   A. Dacă produsul elementelor de pe prima coloană este mai mare decât valoarea lui *k*, declarată ca o constantă globală, atunci să se sorteze crescător elementele de pe diagonala principală a array-ului, aplicând tehnica de sortare QuickSort, altfel să

se sorteze descrescător elementele de pe diagonala secundară a array-ului, aplicând tehnica de sortare ShellSort.

B. Dacă elementul minim din array se află cel puțin o dată pe linie impară, atunci să se sorteze crescător elementele liniei în care se află elementul minim, aplicând tehnica de sortare SelectionSort, altfel să se sorteze descrescător elementele de pe coloana în care se află elementul minim, aplicând tehnica de sortare InsertionSort.

Varianta 16

1. Se dă un array unidimensional cu elemente de tip *integer* și un număr impar *n* (condiția dată trebuie să se verifice în program!), valoarea căruia este citită de la tastatură. Să se afișeze array-ul original și array-ul modificat după fiecare manipulare a datelor din array, afișarea rezultatelor ca și concluzie (De ex.: Nu există numere prime; Nu există așa element cu indexul dat și alte date stipulate în condiția problemei) etc.

   A. Dacă există cel puțin un element nul, atunci să se sorteze elementele array-ului descrescător, aplicând tehnica de sortare HeapSort, altfel să se sorteze elementele array-ului crescător, aplicând tehnica de sortare CountingSort.

   B. Dacă produsul elementelor impare a array-ului este un număr pozitiv (array-ul va conține numere atât pozitive, cât și negative), atunci să se sorteze elementele array-ului crescător, aplicând tehnica de sortare RadixSort, altfel să se sorteze elementele array-ului descrescător, aplicând tehnica de sortare CombSort.

   C. Dacă există cel puțin un element număr prim, atunci să se sorteze elementele array-ului descrescător, aplicând tehnica de sortare MergeSort, altfel să se sorteze elementele array-ului crescător, aplicând tehnica de sortare BubbleSort.

2. Se dă un array bidimensional cu elemente de tip *integer* și un număr natural n ($n \geq 0$), valoarea căruia este citită de la tastatură. Să se afișeze array-ul original și array-ul modificat după fiecare manipulare a datelor din array, afișarea rezultatelor ca și concluzie (De ex.: Nu există numere prime; Nu există așa element cu indexul dat și alte date stipulate în condiția problemei) etc.

   A. Dacă produsul elementelor de sub diagonala principală este mai mare decât valoarea lui *k*, declarată ca o constantă globală, atunci să se sorteze crescător elementele de pe diagonala principală a array-ului, aplicând tehnica de sortare QuickSort, altfel să se sorteze descrescător elementele de pe diagonala secundară a array-ului, aplicând tehnica de sortare ShellSort.

   B. Dacă elementul minim din array se întâlnește numai o singură dată în array, atunci să se sorteze crescător elementele liniei în care se află elementul minim, aplicând tehnica de sortare SelectionSort, altfel să se sorteze descrescător elementele de pe coloanele în care se află elementul minim, aplicând tehnica de sortare InsertionSort.

Varianta 17