#include <stdio.h>

#include <stdlib.h>

#include <stdbool.h>

#include <time.h>

void meniu(int \*\*\*a, int \*\*arr, int n);

void citire(int \*\*a, int n);

void alocare(int \*\*\*a, int n);

void afisare(int \*\*a, int n);

void random\_arr(int \*\*a, int n);

void sortare\_bubble1Dcres(int \*arr, int n);

void sortare\_selection1Dcres(int \*arr, int n);

void sortare\_insertion1Dcres(int \*arr, int n);

int sumaCifrelor(int numar);

void fornew1D(int \*\*a, int \*arr, int k, int index, int n);

void freemem2D(int \*\*a, int n);

void freemem1D(int \*arr);

void afisare1D(int \*arr, int index);

void sortare\_bubble1Ddesc(int \*arr, int n);

void sortare\_selection1Ddesc(int \*arr, int n);

void sortare\_insertion1Ddesc(int \*arr, int n);

int main() {

    int n, k, numar, \*\*a = NULL;

    printf("----------Introduceti dimensiunile la ARRAY!!!------- : ");

    printf("Dati nr de linii si coloane n:");

    scanf("%i", &n);

    printf("Dati suma numerilor k:");

    scanf("%i", &k);

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

    int index = 0;

    meniu(&a, &arr, n);

    return 0;

}

void meniu(int \*\*\*a, int \*\*arr, int n) {

    int option;

    printf("\n--------Lista de optiuni--------\n"

           "1. Alocarea dinamica a memoriei\n"

           "2. Introducerea elementelor tabloului de la tastatura\n"

           "3. Elemente random pentru 2D\n"

           "4. Sortarea elementelor tabloului Bubble sort crescator\n"

           "5. Sortarea elementelor tabloului Bubble sort descrescator\n"

           "6. Sortarea elementelor tabloului Selection sort crescator\n"

           "7. Sortarea elementelor tabloului Selection sort descrescator\n"

           "8. Sortarea elementelor tabloului Insertion sort crescator\n"

           "9. Sortarea elementelor tabloului Insertion sort descrescator\n"

           "10. Afisarea elementelor tabloului la ecran 2D\n"

           "11. Afisarea elementelor tabloului la ecran 1D\n"

           "0. Iesire din program\n");

    printf("Selectati si introduceti nr optiunii dorite: ");

    scanf("%i", &option);

    switch (option) {

        case 0: {

            printf("Se efectueaza iesire din program...");

            freemem2D(\*a, n);

            freemem1D(\*arr);

            exit(1);

            break;

        }

        case 1: {

            alocare(a, n);

            break;

        }

        case 2: {

            citire(\*a, n);

            meniu(a, arr, n);

            break;

        }

        case 3: {

            random\_arr(\*a, n);

            meniu(a, arr, n);

            break;

        }

        case 4: {

            sortare\_bubble1Dcres(\*arr, n \* n);

            meniu(a, arr, n);

            break;

        }

        case 5: {

            sortare\_bubble1Ddesc(\*arr, n \* n);

            meniu(a, arr, n);

            break;

        }

        case 6: {

            sortare\_selection1Dcres(\*arr, n \* n);

            meniu(a, arr, n);

            break;

        }

        case 7: {

            sortare\_selection1Ddesc(\*arr, n \* n);

            meniu(a, arr, n);

            break;

        }

        case 8: {

            sortare\_insertion1Dcres(\*arr, n \* n);

            meniu(a, arr, n);

            break;

        }

        case 9: {

            sortare\_insertion1Ddesc(\*arr, n \* n);

            meniu(a, arr, n);

            break;

        }

        case 10: {

            afisare(\*a, n);

            break;

        }

        case 11: {

            afisare1D(\*arr, n \* n);

            break;

        }

        default: {

            printf("\n-----------Nu exista optiunea!!!-----------\n");

            meniu(a, arr, n);

        }

    }

}

void alocare(int \*\*\*a, int n) {

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

    for (int i = 0; i < n; i++) {

        (\*a)[i] = (int \*)calloc(n, sizeof(int));

    }

    if (\*a == NULL) {

        printf(" Eroare. Nu s-a alocat memorie.\n");

        exit(1);

    } else {

        printf("Succes.\n");

        meniu(a, arr, n);

    }

}

void citire(int \*\*a, int n) {

    for (int i = 0; i < n; i++) {

        for (int j = 0; j < n; j++) {

            scanf("%i", &a[i][j]);

        }

    }

    meniu(a, arr, n);

}

void afisare(int \*\*a, int n) {

    for (int i = 0; i < n; i++) {

        for (int j = 0; j < n; j++) {

            printf("%i", a[i][j]);

            printf("\n");

        }

    }

}

void random\_arr(int \*\*a, int n) {

    srand(time(NULL));

    for (int i = 0; i < n; i++) {

        for (int j = 0; j < n; j++) {

            a[i][j] = rand() % 100;

        }

    }

    printf("Tabloul a fost completat cu valori aleatorii <<100%%>>");

    meniu(a, arr, n);

}

void sortare\_bubble1Dcres(int \*arr, int n) {

    int aux;

    for (int i = 0; i < n; i++) {

        for (int j = 0; j < n - i - 1; j++) {

            if (arr[j] > arr[j + 1]) {

                aux = arr[j];

                arr[j] = arr[j + 1];

                arr[j + 1] = aux;

            }

        }

    }

    printf("\n------------Tablou sortat <<100%%>>--------\n");

}

void sortare\_insertion1Dcres(int \*arr, int n) {

    int key, j;

    for (int i = 1; i < n; i++) {

        key = arr[i];

        j = i - 1;

        while (j >= 0 && arr[j] > key) {

            arr[j + 1] = arr[j];

            j = j - 1;

        }

        arr[j + 1] = key;

    }

}

void sortare\_selection1Dcres(int \*arr, int n) {

    int j, minIndex, temp;

    for (int i = 0; i < n - 1; ++i) {

        minIndex = i;

        for (j = i + 1; j < n; ++j) {

            if (arr[j] < arr[minIndex]) {

                minIndex = j;

            }

        }

        temp = arr[minIndex];

        arr[minIndex] = arr[i];

        arr[i] = temp;

    }

}

int sumaCifrelor(int numar) {

    int suma = 0;

    while (numar != 0) {

        suma += numar % 10;

        numar /= 10;

    }

    return suma;

}

void fornew1D(int \*\*a, int \*arr, int k, int index, int n) {

    for (int i = 0; i < n; ++i) {

        for (int j = 0; j < n; ++j) {

            if (sumaCifrelor(a[i][j]) == k) {

                arr[index++] = a[i][j];

            }

        }

    }

}

void freemem2D(int \*\*a, int n) {

    for (int i = 0; i < n; i++)

        free(a[i]);

    free(a);

}

void afisare1D(int \*arr, int index) {

    for (int i = 0; i < index; i++) {

        printf("%i", arr[i]);

    }

}

void freemem1D(int \*arr) {

    free(arr);

}

void sortare\_bubble1Ddesc(int \*arr, int n) {

    int aux;

    for (int i = 0; i < n; i++) {

        for (int j = 0; j < n - i - 1; j++) {

            if (arr[j] < arr[j + 1]) {

                aux = arr[j];

                arr[j] = arr[j + 1];

                arr[j + 1] = aux;

            }

        }

    }

}

void sortare\_selection1Ddesc(int \*arr, int n) {

    int maxIndex, temp;

    for (int i = 0; i < n - 1; ++i) {

        maxIndex = i;

        for (int j = i + 1; j < n; ++j) {

            if (arr[j] > arr[maxIndex]) {

                maxIndex = j;

            }

        }

        temp = arr[maxIndex];

        arr[maxIndex] = arr[i];

        arr[i] = temp;

    }

}

void sortare\_insertion1Ddesc(int \*arr, int n) {

    int key, j;

    for (int i = 1; i < n; i++) {

        key = arr[i];

        j = i - 1;

        while (j >= 0 && arr[j] < key) {

            arr[j + 1] = arr[j];

            j = j - 1;

        }

        arr[j + 1] = key;

    }

}

#include <stdio.h>

#include <stdlib.h>

#include <stdbool.h>

#include <time.h>

void meniu(int \*\*\*a, int \*\*arr, int n);

void citire(int \*\*a, int n);

void alocare(int \*\*\*a, int n);

void afisare(int \*\*a, int n);

void random\_arr(int \*\*a, int n);

void sortare\_bubble1Dcres(int \*arr, int n);

void sortare\_selection1Dcres(int \*arr, int n);

void sortare\_insertion1Dcres(int \*arr, int n);

int sumaCifrelor(int numar);

void fornew1D(int \*\*a, int \*arr, int k, int index, int n);

void freemem2D(int \*\*a, int n);

void freemem1D(int \*arr);

void afisare1D(int \*arr, int index);

void sortare\_bubble1Ddesc(int \*arr, int n);

void sortare\_selection1Ddesc(int \*arr, int n);

void sortare\_insertion1Ddesc(int \*arr, int n);

int main() {

    int n, k, numar, \*\*a = NULL;

    printf("----------Introduceti dimensiunile la ARRRAY!!!------- : ");

    printf("Dati nr de linii si coloane n:");

    scanf("%i", &n);

    printf("Dati suma numerilor k:");

    scanf("%i", &k);

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

    int index = 0;

    meniu(&a, &arr, n);

    return 0;

}

void meniu(int \*\*\*a, int \*\*arr, int n) {

    int option;

    printf("\n--------Lista de optiuni--------\n"

           "1. Alocarea dinamica a memoriei\n"

           "2. Introducerea elementelor tabloului de la tastatura\n"

           "3. Elemente random pentru 2D\n"

           "4. Sortarea elementelor tabloului Bubble sort crescator\n"

           "5. Sortarea elementelor tabloului Bubble sort descrescator\n"

           "6. Sortarea elementelor tabloului Selection sort crescator\n"

           "7. Sortarea elementelor tabloului Selection sort descrescator\n"

           "8. Sortarea elementelor tabloului Insertion sort crescator\n"

           "9. Sortarea elementelor tabloului Insertion sort descrescator\n"

           "10. Afisarea elementelor tabloului la ecran 2D\n"

           "11. Afisarea elementelor tabloului la ecran 1D\n"

           "0. Iesire din program\n");

    printf("Selectati si introduceti nr optiunii dorite: ");

    scanf("%i", &option);

    switch (option) {

        case 0: {

            printf("Se efectueaza iesire din program...");

            freemem2D(\*a, n);

            freemem1D(\*arr);

            exit(1);

            break;

        }

        case 1: {

            alocare(a, n);

            break;

        }

        case 2: {

            citire(\*a, n);

            meniu(a, arr, n);

            break;

        }

        case 3: {

            random\_arr(\*a, n);

            meniu(a, arr, n);

            break;

        }

        case 4: {

            sortare\_bubble1Dcres(\*arr, n \* n);

            meniu(a, arr, n);

            break;

        }

        case 5: {

            sortare\_bubble1Ddesc(\*arr, n \* n);

            meniu(a, arr, n);

            break;

        }

        case 6: {

            sortare\_selection1Dcres(\*arr, n \* n);

            meniu(a, arr, n);

            break;

        }

        case 7: {

            sortare\_selection1Ddesc(\*arr, n \* n);

            meniu(a, arr, n);

            break;

        }

        case 8: {

            sortare\_insertion1Dcres(\*arr, n \* n);

            meniu(a, arr, n);

            break;

        }

        case 9: {

            sortare\_insertion1Ddesc(\*arr, n \* n);

            meniu(a, arr, n);

            break;

        }

        case 10: {

            afisare(\*a, n);

            break;

        }

        case 11: {

            afisare1D(\*arr, n \* n);

            break;

        }

        default: {

            printf("\n-----------Nu exista optiunea!!!-----------\n");

            meniu(a, arr, n);

        }

    }

}

void alocare(int \*\*\*a, int n) {

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

    for (int i = 0; i < n; i++) {

        (\*a)[i] = (int \*)calloc(n, sizeof(int));

    }

    if (\*a == NULL) {

        printf(" Eroare. Nu s-a alocat memorie.\n");

        exit(1);

    } else {

        printf("Succes.\n");

        meniu(a, arr, n);

    }

}

void citire(int \*\*a, int n) {

    for (int i = 0; i < n; i++) {

        for (int j = 0; j < n; j++) {

            scanf("%i", &a[i][j]);

        }

    }

    meniu(a, arr, n);

}

void afisare(int \*\*a, int n) {

    for (int i = 0; i < n; i++) {

        for (int j = 0; j < n; j++) {

            printf("%i", a[i][j]);

            printf("\n");

        }

    }

}

void random\_arr(int \*\*a, int n) {

    srand(time(NULL));

    for (int i = 0; i < n; i++) {

        for (int j = 0; j < n; j++) {

            a[i][j] = rand() % 100;

        }

    }

    printf("Tabloul a fost completat cu valori aleatorii <<100%%>>");

    meniu(a, arr, n);

}

void sortare\_bubble1Dcres(int \*arr, int n) {

    int aux;

    for (int i = 0; i < n; i++) {

        for (int j = 0; j < n - i - 1; j++) {

            if (arr[j] > arr[j + 1]) {

                aux = arr[j];

                arr[j] = arr[j + 1];

                arr[j + 1] = aux;

            }

        }

    }

    printf("\n------------Tablou sortat <<100%%>>--------\n");

}

void sortare\_insertion1Dcres(int \*arr, int n) {

    int key, j;

    for (int i = 1; i < n; i++) {

        key = arr[i];

        j = i - 1;

        while (j >= 0 && arr[j] > key) {

            arr[j + 1] = arr[j];

            j = j - 1;

        }

        arr[j + 1] = key;

    }

}

void sortare\_selection1Dcres(int \*arr, int n) {

    int j, minIndex, temp;

    for (int i = 0; i < n - 1; ++i) {

        minIndex = i;

        for (j = i + 1; j < n; ++j) {

            if (arr[j] < arr[minIndex]) {

                minIndex = j;

            }

        }

        temp = arr[minIndex];

        arr[minIndex] = arr[i];

        arr[i] = temp;

    }

}

int sumaCifrelor(int numar) {

    int suma = 0;

    while (numar != 0) {

        suma += numar % 10;

        numar /= 10;

    }

    return suma;

}

void fornew1D(int \*\*a, int \*arr, int k, int index, int n) {

    for (int i = 0; i < n; ++i) {

        for (int j = 0; j < n; ++j) {

            if (sumaCifrelor(a[i][j]) == k) {

                arr[index++] = a[i][j];

            }

        }

    }

}

void freemem2D(int \*\*a, int n) {

    for (int i = 0; i < n; i++)

        free(a[i]);

    free(a);

}

void afisare1D(int \*arr, int index) {

    for (int i = 0; i < index; i++) {

        printf("%i", arr[i]);

    }

}

void freemem1D(int \*arr) {

    free(arr);

}

void sortare\_bubble1Ddesc(int \*arr, int n) {

    int aux;

    for (int i = 0; i < n; i++) {

        for (int j = 0; j < n - i - 1; j++) {

            if (arr[j] < arr[j + 1]) {

                aux = arr[j];

                arr[j] = arr[j + 1];

                arr[j + 1] = aux;

            }

        }

    }

}

void sortare\_selection1Ddesc(int \*arr, int n) {

    int maxIndex, temp;

    for (int i = 0; i < n - 1; ++i) {

        maxIndex = i;

        for (int j = i + 1; j < n; ++j) {

            if (arr[j] > arr[maxIndex]) {

                maxIndex = j;

            }

        }

        temp = arr[maxIndex];

        arr[maxIndex] = arr[i];

        arr[i] = temp;

    }

}

void sortare\_insertion1Ddesc(int \*arr, int n) {

    int key, j;

    for (int i = 1; i < n; i++) {

        key = arr[i];

        j = i - 1;

        while (j >= 0 && arr[j] < key) {

            arr[j + 1] = arr[j];

            j = j - 1;

        }

        arr[j + 1] = key;

    }

}

#include <stdio.h>

#include <stdlib.h>

#include <stdbool.h>

#include <time.h>

void meniu(int \*\*a, int \*arr, int n);

void citire(int \*\*a,int \*arr, int n);

void alocare(int \*\*a,int \*arr, int n);

void afisare(int \*\*a,int \*arr, int n);

void random\_arr(int \*\*a,int \*arr, int n);

void sortare\_bubble1Dcres(int \*\*a, int \*arr, int n);

void sortare\_selection1Dcres(int \*\*a, int \*arr, int n);

void sortare\_insertion1Dcres(int \*\*a, int \*arr, int n);

int sumaCifrelor(int numar);

void fornew1D(int \*\*a, int \*arr, int n);

void freemem2D(int \*\*a,int \*arr, int n);

void freemem1D(int \*\*a, int \*arr, int n);

void afisare1D(int \*\*a, int \*arr, int index);

void sortare\_bubble1Ddesc(int \*\*a, int \*arr, int n);

void sortare\_selection1Ddesc(int \*\*a, int \*arr, int n);

void sortare\_insertion1Ddesc(int \*\*a, int \*arr, int n);

void alocare1D(int \*\*a, int \*arr, int n);

int main() {

    int n, k, numar, \*\*a = NULL, \*arr=NULL;

    printf("----------Introduceti dimensiunile la ARRRAY!!!------- :\n ");

    printf("Dati nr de linii si coloane n:\n");

    scanf("%i", &n);

    printf("Dati suma numerelor k:\n");

    scanf("%i", &k);

    meniu(a, arr, n);

    return 0;

}

void meniu(int \*\*a, int \*arr, int n) {

    int option;

    printf("\n--------Lista de optiuni--------\n"

           "1. Alocarea dinamica a memoriei\n"

           "2. Introducerea elementelor tabloului de la tastatura\n"

           "3. Elemente random pentru 2D\n"

           "4. Sortarea elementelor tabloului Bubble sort crescator\n"

           "5. Sortarea elementelor tabloului Bubble sort descrescator\n"

           "6. Sortarea elementelor tabloului Selection sort crescator\n"

           "7. Sortarea elementelor tabloului Selection sort descrescator\n"

           "8. Sortarea elementelor tabloului Insertion sort crescator\n"

           "9. Sortarea elementelor tabloului Insertion sort descrescator\n"

           "10. Afisarea elementelor tabloului la ecran 2D\n"

           "11. Afisarea elementelor tabloului la ecran 1D\n"

           "12. Alocarea dinamica a memoriei vectorului 1D\n"

           "13, Calcularea sumei\n"

           "0. Iesire din program\n");

    printf("Selectati si introduceti nr optiunii dorite: ");

    scanf("%i", &option);

    switch (option) {

        case 0: {

            printf("Se efectueaza iesire din program...");

            freemem2D(a, arr, n);

            freemem1D(a, arr, n);

            break;

        }

        case 1: {

            alocare(a, arr, n);

            break;

        }

        case 2: {

            citire(a, arr, n);

            break;

        }

        case 3: {

            random\_arr(a, arr, n);

            break;

        }

        case 4: {

            sortare\_bubble1Dcres(a, arr, n);

            meniu(a, arr, n);

            break;

        }

        case 5: {

            sortare\_bubble1Ddesc(a, arr, n);

            meniu(a, arr, n);

            break;

        }

        case 6: {

            sortare\_selection1Dcres(a, arr, n);

            meniu(a, arr, n);

            break;

        }

        case 7: {

            sortare\_selection1Ddesc(a, arr, n);

            meniu(a, arr, n);

            break;

        }

        case 8: {

            sortare\_insertion1Dcres(a, arr, n);

            meniu(a, arr, n);

            break;

        }

        case 9: {

            sortare\_insertion1Ddesc(a, arr, n);

            meniu(a, arr, n);

            break;

        }

        case 10: {

            afisare(a, arr, n);

            break;

        }

        case 11: {

            afisare1D(a, arr, n);

            break;

        }

        case 12:

           alocare1D(a, arr, n);

           break;

        case 13:

           fornew1D(a, arr, n);

           break;

        default: {

            printf("\n-----------Nu exista optiunea!!!-----------\n");

            meniu(a, arr, n);

        }

    }

}

void alocare(int \*\*a,int \*arr, int n) {

    a = malloc(n \* sizeof(int \*));

    for (int i = 0; i < n; i++) {

        a[i] = calloc(n, sizeof(int));

    }

    if (a == NULL) {

        printf(" Eroare. Nu s-a alocat memorie.\n");

        exit(1);

    } else {

        printf("Succes.\n");

        meniu(a, arr, n);

    }

}

void alocare1D(int \*\*a, int \*arr, int n){

   arr = malloc(n \* sizeof(int \*));

   for(int i = 0; i < n; i++){

        arr[i] = calloc(n, sizeof(int));

   }

   if (arr == NULL) {

       printf("Eroare. Nu s-a alocat memorie.\n");

       exit(1);

     } else {

        printf("Succes.\n");

       meniu(a, arr, n);

  }

}

void citire(int \*\*a, int \*arr, int n) {

    for (int i = 0; i < n; i++) {

        for (int j = 0; j < n; j++) {

            scanf("%i", &a[i][j]);

        }

    }

    meniu(a, arr, n);

}

void afisare(int \*\*a, int \*arr, int n) {

    for (int i = 0; i < n; i++) {

        for (int j = 0; j < n; j++) {

            printf("%i", a[i][j]);

        }

        printf("\n");

    }

    meniu(a, arr, n);

}

void random\_arr(int \*\*a, int \*arr, int n) {

    srand(time(NULL));

    for (int i = 0; i < n; i++) {

        for (int j = 0; j < n; j++) {

            a[i][j] = rand() % 100;

        }

    }

    printf("Tabloul a fost completat cu valori aleatorii <<100%%>>");

    meniu(a, arr, n);

}

void sortare\_bubble1Dcres(int \*\*a, int \*arr, int n) {

    int aux;

    for (int i = 0; i < n; i++) {

        for (int j = 0; j < n - i - 1; j++) {

            if (arr[j] > arr[j + 1]) {

                aux = arr[j];

                arr[j] = arr[j + 1];

                arr[j + 1] = aux;

            }

        }

    }

     afisare1D(a, arr, n);

      meniu(a, arr, n);

}

void sortare\_insertion1Dcres(int \*\*a, int \*arr, int n) {

    int key, j;

    for (int i = 1; i < n; i++) {

        key = arr[i];

        j = i - 1;

        while (j >= 0 && arr[j] > key) {

            arr[j + 1] = arr[j];

            j = j - 1;

        }

        arr[j + 1] = key;

    }

   afisare1D(a, arr, n);

    meniu(a, arr, n);

}

void sortare\_selection1Dcres(int \*\*a, int \*arr, int n) {

    int minIndex, temp;

    for (int i = 0; i < n - 1; ++i) {

        minIndex = i;

        for (int j = i + 1; j < n; ++j) {

            if (arr[j] < arr[minIndex]) {

                minIndex = j;

        temp = arr[minIndex];

        arr[minIndex] = arr[i];

        arr[i] = temp;

            }

        }

    }

    afisare1D(a, arr, n);

     meniu(a, arr, n);

}

int sumaCifrelor(int numar) {

    int suma = 0;

    while (numar != 0) {

        suma += numar % 10;

        numar /= 10;

    }

    return suma;

}

void fornew1D(int \*\*a, int \*arr, int n) {

   int index = 0, k;

    for (int i = 0; i < n; ++i) {

        for (int j = 0; j < n; ++j) {

            if (sumaCifrelor(a[i][j]) == k) {

                arr[index++] = a[i][j];

            }

        }

    }

    afisare1D(a, arr, n);

    meniu(a, arr, n);

}

void freemem2D(int \*\*a, int \*arr, int n) {

    for (int i = 0; i < n; i++)

        free(a[i]);

    free(a);

}

void afisare1D(int \*\*a, int \*arr, int n) {

    for (int i = 0; i < n; i++) {

        printf("%i", arr[i]);

    }

    printf("\n");

    meniu(a, arr, n);

}

void freemem1D(int \*\*a, int \*arr, int n) {

    free(arr);

}

void sortare\_bubble1Ddesc(int \*\*a, int \*arr, int n) {

    int aux;

    for (int i = 0; i < n; i++) {

        for (int j = 0; j < n - i - 1; j++) {

            if (arr[j] < arr[j + 1]) {

                aux = arr[j];

                arr[j] = arr[j + 1];

                arr[j + 1] = aux;

            }

        }

    }

    afisare1D(a, arr, n);

     meniu(a, arr, n);

}

void sortare\_selection1Ddesc(int \*\*a, int \*arr, int n) {

    int maxIndex, temp;

    for (int i = 0; i < n - 1; ++i) {

        maxIndex = i;

        for (int j = i + 1; j < n; ++j) {

            if (arr[j] > arr[maxIndex]) {

                maxIndex = j;

        temp = arr[maxIndex];

        arr[maxIndex] = arr[i];

        arr[i] = temp;

            }

        }

    }

    afisare1D(a, arr, n);

     meniu(a, arr, n);

}

void sortare\_insertion1Ddesc(int \*\*a, int \*arr, int n) {

    int key, j;

    for (int i = 1; i < n; i++) {

        key = arr[i];

        j = i - 1;

        while (j >= 0 && arr[j] < key) {

            arr[j + 1] = arr[j];

            j = j - 1;

        }

        arr[j + 1] = key;

    }

    afisare1D(a, arr, n);

    meniu(a, arr, n);

}