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. *Alocarea dinamică a memoriei;*
2. *Introducerea valorilor tabloului de la tastatură;*
3. *Completarea tabloului cu valori random;*
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.*

```
#include <stdio.h>
#include <time.h>
#include <stdlib.h>
#include <stdbool.h>

void meniu(int **a, int n, int m);
void alocare (int **a, int n, int m);
void citire(int **a, int n, int m);
void afisare(int **a, int n, int m);
void random_arr(int **a, int n, int m);
void sortare(int **a, int n, int m);
void sortare_arr(int **a, int n, int m);
void freemem(int **a, int n, int m);
```

```
int main()
{
   int **arr = NULL, n, m;

   printf("Introduceti dimensiunile tabloului:\n");
   printf("Dati nr de linii: ");
   scanf("%i", &n);
   printf("Dati nr de coloane: ");
   scanf("%i", &m);

   meniu(arr, n, m);
 return 0;
}
```

```
void meniu(int **a, int n, int m){
    int option;
    printf("\nLista de optiuni:\n"
          "1. Alocarea dinamica a memoriei\n"
```

```
          "2. Introducerea elementelor tabloului de la tastatura\n"
          "3. Completarea tabloului cu valori aleatorii\n"
          "41. Sortarea elementelor tabloului pe linie\n"
          "42. Sortarea elementelor tabloului\n"
          "5. Afisarea elementelor tabloului la ecran\n"
          "0. Iesire din program\n");
   printf("Selectati si introduceti nr optiunii dorite: ");
   scanf("%i", &option);

  switch (option)
    {
      case 0:
         printf("Se efectueaza iesirea din program...");
         freemem(a, n, m);
         exit(1);
         break;
      case 1:
         alocare(a, n, m);
         break;
      case 2:
         citire(a, n, m); //tastatura
         break;
      case 3:
         random_arr(a, n, m);  //aleatoriu
         break;
      case 41:
         sortare(a, n, m);  //bubble sort pe linie
         meniu(a, n, m);
         break;
      case 42:
         sortare_arr(a, n, m);  //bubble sort
         meniu(a, n, m);
         break;
      case 5:
         afisare(a, n, m);
         break;
      default:
         {printf("\n-----------Nu exista optiune cu asa numar!-----------\n");
         meniu(a, n, m);
         }
      }
    }
void alocare(int **a, int n, int m){
    a =  malloc (n * sizeof(int *));
    for(int i = 0; i < n; i++){
        a[i] = calloc(m, sizeof(int));
    }
    if (a == NULL){
        printf("Eroare. Nu s-a alocat memorie.\n");
        exit(1);
    } else {
        printf("Succes.\n");
        meniu(a, n, m);
    }
}
```

```
void citire(int **a, int n, int m){
    int i, j;
    for(i = 0; i < n; i++)
    for (j = 0; j < m; j++)
    scanf("%i", &a[i][j]);
    meniu(a, n, m);
}
```

```
void afisare(int **a, int n, int m){
        int i, j;
        for(i = 0; i < n; i++)
        {
            for (j = 0; j < m; j++)
            {
                printf("%i\t", a[i][j]);
            }
            printf("\n");
        }
        meniu(a, n, m);
}
```

```
void random_arr(int **a, int n, int m){
   srand(time(NULL));
   for (int i = 0; i < n; i++)
   {
      for (int j = 0; j < m; j++)
      {
         a[i][j] = rand() % 100;
      }
   }
   printf("--------Tabloul a fost completat cu valori aleatorii.----------\n");
   meniu(a, n, m);
}
```

```
void sortare(int **a, int n, int m){
    int i, j, aux;
    bool sortat;
    do
    {
        sortat = true;
        for(i = 0 ; i < n ; i ++)
        for(j = 0 ; j < m -1 ; j ++)
            if(a[i][j] > a[i][j+1])
            {
                aux = a[i][j];
                a[i][j] = a[i][j+1];
                a[i][j+1] = aux;
                sortat = false;
            }
    } while(!sortat);
    printf("\n--------Tabloul a fost sortat pe linie.----------\n");
}
```

```
void sortare_arr(int **a, int n, int m){
    int i, j, aux, v[n*m], k = 0;
```

```
    bool sortat;
    do
    {
        sortat = true;
        for(i = 0 ; i < n ; i ++){
        for(j = 0 ; j < m -1 ; j ++){
            if(a[i][j] > a[i][j+1])
            {
                aux = a[i][j];
                a[i][j] = a[i][j+1];
                a[i][j+1] = aux;
                sortat = false;
            }
        }
            if((i < n-1) && (a[i][m-1] > a[i+1][0]))
            {
                aux = a[i][m-1];
                a[i][m-1] = a[i+1][0];
                a[i+1][0] = aux;
                sortat = false;
            }
        }
    } while(!sortat);
/*    for(i = 0 ; i < n ; i ++){
        for(j = 0 ; j < m ; j ++){
        v[k++] = a[i][j];
        }
    }
    do
    {
    sortat = true;
    for(i = 0 ; i < n*m - 1 ; i ++)
        if(v[i] > v[i+1])
        {
            int aux = v[i];
            v[i] = v[i+1];
            v[i+1] = aux;
            sortat = false;
        }
    } while(!sortat);
    k = 0;
    for(i = 0 ; i < n ; i ++){
        for(j = 0 ; j < m ; j ++){
            a[i][j] = v[k++];
        }
    }*/
    printf("\n--------Tabloul a fost sortat.----------\n");
}
```

```
void freemem(int **a, int n, int m){
        int i;
        for (i = 0; i < n; i++)
        free(a[i]);
        free(a);
}
```