22.02.2022

PROGRAMARE PROCEDURALĂ
– LABORATOR NR. 2 –

1. Scrieți o funcție care să returneze un tablou unidimensional alocat dinamic format din valorile strict mai mici decât o valoare x aflate între doi indici i și j $(0 \leq i \leq j < n)$ într-un tablou unidimensional v având n elemente de tip întreg, precum și numărul acestora. Folosind apeluri utile ale funcției definite anterior, scrieți o funcție care să afișeze mesajul DA în cazul în care un tabloul unidimensional v format din n numere este sortat crescător sau mesajul NU în caz contrar.

Exemplu:

```
Fie v = (2, 10, 7, 5, 2, 1, 13, 4, 14, 7, 5), n = 11, x = 6, i = 2, j = 8

void ExtragereValori(int *v, int x, int i, int j, int **vsm, int *nsm)

*vsm = (2, 1, 4), *nsm = 0 1 2 3
.................................

int *tsm, sm;
ExtragereValori(v, x, i, j, &tsm, &sm); => tsm = (2, 1, 4), sm = 3
.................................
```

Varianta 1:

```
Fie v = (2, 5, 7, 10, 13, 14), n = 6
```

Verific dacă `v[i] <= v[i+1]` pentru orice indice `i` cuprins între `0` și `n-2`:

```
.................................
for(i = 0; i < n-1; i++)
{
   ExtragereValori(v, v[i], i, i+1, &tsm, &sm);
   free(tsm);
   if(sm != 0)
   {
         printf("NU");
         return;
   }
}
printf("DA");
```

Varianta 2:

Fie v = (2, 5, 7, 10, 14, 13), n = 6

Verific dacă v[i] <= min{v[i+1],…,v[n-1]} pentru orice indice i cuprins între 0 și n-2, adică nu există nicio valoare strict mai mică decât v[i] în secvența v[i+1],…,v[n-1]:

```
.................................
for(i = 0; i < n-1; i++)
{
   ExtragereValori(v, v[i], i+1, n-1, &tsm, &sm);
   free(tsm);
   if(sm != 0)
   {
         printf("NU");
         return;
   }
}
printf("DA");
```

Rezolvare:

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

void ExtragereValori(int *v, int x, int i, int j, int **vsm, int *nsm)
{
    int k,p;

    *nsm = 0;
    for(k = i; k <= j; k++)
        if(v[k] < x)
            (*nsm)++;

    *vsm = malloc(*nsm * sizeof(int));

    p = 0;
    for(k = i; k <= j; k++)
        if(v[k] < x)
            (*vsm)[p++] = v[k];
}

void afisare(int *v, int n)
{
    int i;

    for (i = 0; i < n; i++)
        printf("%d ", v[i]);
```

```
    printf("\n");
}

void citire(int **v,int *n)
{
    int i;

    printf("n = ");
    scanf("%d", n);

    *v = malloc(*n*sizeof(int));

    printf("Introduceti elementele tabloului: \n");

    for(i = 0; i < *n; i++)
        scanf("%d", *v+i);

}

void sortat(int *v, int n)
{
    int i,*tsm,sm;

    for(i = 0; i <= n-2; i++)
    {
        ExtragereValori(v, v[i], i+1, n-1, &tsm, &sm);
        free(tsm);
        if(sm != 0)
        {
            printf("NU");
            return;
        }
    }
    printf("DA");
}

int main()
{
    int *v, n;

    citire(&v,&n);

    sortat(v,n);
    free(v);

    return 0;
}
```

2. Scrieți o funcție care să returneze un tablou unidimensional alocat dinamic format din valorile strict pozitive aflate între doi indici i și j ($0 \leq i \leq j < n$) într-un tablou unidimensional v având n elemente de tip întreg, precum și numărul acestora. Folosind apeluri utile ale funcției definite anterior, scrieți o funcție care să afișeze mesajul DA în cazul în care un tabloul unidimensional v având n elemente numere întregi nenule ($n \geq 2$ număr natural par) are prima jumătate formată doar din numere strict negative și a doua jumătate doar din numere strict pozitive, respectiv mesajul NU în caz contrar.

Rezolvare:

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

void ExtragereValori(int *v, int i, int j, int **vsp, int *nsp)
{
    int k, *aux;

    *nsp = 0;
    *vsp = NULL;
    for(k = i; k <= j; k++)
        if(v[k] > 0)
        {
            (*nsp)++;
            aux = realloc(*vsp, *nsp * sizeof(int));
            if (aux == NULL)
            {
                printf("\nMemorie insuficenta\n");
                free(*vsp);
                *vsp = NULL;
                *nsp = 0;
                return;
            }
            *vsp = aux;
            (*vsp)[*nsp - 1] = v[k];
        }
}

void afisare(int *v, int n)
{
    int i;

    for (i = 0; i < n; i++)
        printf("%d ", v[i]);
    printf("\n");
}
```

```
void citire(int **v,int *n)
{
    int i;

    printf("n = ");
    scanf("%d", n);

    *v = malloc(*n*sizeof(int));

    printf("Introduceti elementele tabloului: \n");

    for(i = 0; i < *n; i++)
        scanf("%d", *v+i);

}

void verificare(int *v, int n)
{
    int *jumate, npoz, nneg;

    ExtragereValori(v, 0, n/2 - 1, &jumate, &nneg);
    free(jumate);

    ExtragereValori(v, n/2, n - 1, &jumate, &npoz);
    free(jumate);

    if (nneg == 0 && npoz == n / 2)
        printf("DA\n");
    else
        printf("NU\n");
}

int main()
{
    int *v, n;

    citire(&v,&n);

    verificare(v, n);

    free(v);

    return 0;
}
```