Liciu Mihai George CTI 162
11.06.2021

```
#include <stdio.h>
int greedy (int n, int *arr, int negCounter, int negMax)
{ int produs = 1;
  for(int i=0; i<n; i++)
    { if(arr[i]!=0)
      { if(negCounter %2 ==1)
        { if(i!=negMax)
          { produs *=arr[i];
            printf("%d ",arr[i]);
          }
        }
        else
        { produs *= arr[i];
          printf("%d ",arr[i]);
        }
      }
    }
  return produs;
}
int main()
{ int n, negCounter=0, negMax=-1;   //negcounter numără câte numere negative am citit
  scanf("%d",&n);                   // negMax reține indexul celui mai mare număr negativ
  int arr[100000];                  // pe care îl are
  for(int i=0; i<n; i++)
    { scanf("%d",&arr[i]);
      if(arr[i]<0)
        { negCounter ++
          if(negMax==-1||arr[i]>arr[negMax])
            negMax=i;
        }
    }
  printf("\n%d",greedy(n,arr,negCounter,negMax));
  return 0;
}
```

// Programul citește numerele și reține care este cel mai mare număr negativ dintre ele, reținându-i indexul. În cazul acestei probleme nu a fost nevoie de sortare. Apoi, programul parcurge vectorul aflând optimul local pe baza unor condiții și îl înmulțește la produs. Condițiile acestea sunt: numărul să fie nenul, iar dacă numărul de numere negative este impar ne asigurăm că nu îl înmulțim cu cel mai mare număr negativ deoarece am ajunge la un produs final mai mic, în cazul în care numărul de numere negative este par atunci produsul maxim va fi produsul tuturor numerelor nenule. Programul se încadrează în tehnica programării greedy deoarece parcurgem șirul de date pe baza unor condiții pentru a afla un optim local. Complexitatea programului este O(n) deoarece parcurgem liniar datele pentru a le verifica condițiile.