```
int * multiplicaMatrizTransposta (int *v1, int *v2, int n) {

    if(v1 == NULL || v2 == NULL) {
        Return NULL;
    }
    int i, j, k=0, aux, trans;
    for (i=0; i<n, i++) {
        for (j=0; j<n; j++) {
            k = i*n+j;
            trans = j*n+i;
            if(i>j) {
                aux = v2[k];
                v2[k] = v1[trans];
                v2[trans] = aux;
            }
        }
    }
    int *matrizAux = (int*) malloc(sizeof(int)*n);
    if(matrizAux == NULL) {
        Return NULL;
    }
    int l, h, t;
    for(i=0; i<n; i++) {
        for(j=0; j<n; j++) {
            l = i*n+j;
            matrizAux[l] = 0;
            for(k=0; k<n; k++) {
                t = k*n+j;
                h = i*n+k;
                matrizAux[l] += v1[h] * v2[t];
            }
        }
    }
```

```
    }
    return matrizAux;
}
```

```
int qRemoElementosordemImPar(Queue *q){

    if(q==NULL){
        Return false;
    }
    int ele, P, i, aux, cont=0, k=1;
    P=q->front;
    ele= q->elms[q->front];
    P= qIncrCirc(P, q->maxl;
    free(ele);
    for(i=1; i<q->NumElms; i++){
        Aux= q->elms[P];
        if(P!=0){
        q->elms[P-k]=aux;
        }else{
            q->elms[q->max-k]=aux;
        }
        P= qIncrCirc(P, q->max);
        if(i%2==0){
            k=2;
            cont++;
            free(aux);
        }
        k=1;
        q->Numelms--;
    }
    return true;
}
```

```
SLList * sllCopiaInverso (SLList * L1);

    SLList* novoNO = (SLList*) malloc(sizeof(SLList));
    if(novoNo == NULL || L1 == NULL){
       Return NULL;
    }
    No* auxNode = L1-> cabeça;
    No** aux = (No**) malloc(sizeof(No*) * L1->tam);
    if(aux == NULL){
       Return NULL;
    }
    int i, cont = 0;
    for(i=0; i<L1-> tam; i++){
          aux[i] = auxNode;
          auxNode = auxNode -> Prox;
          cont++;
    }
  SLList * auxList = NOVONo;
    for(i= cont; i> cont; i--){
          auxList -> cabeça = Aux[i];
          auxList -> cabeça = AuxList -> cabeça -> Prox;
    }

    AuxList = NovoNo;

    Return Aux List;

}
```