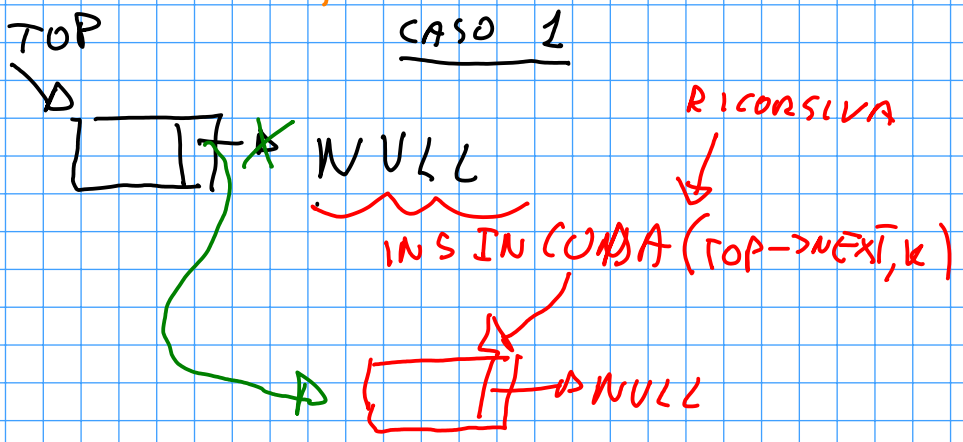
```
IF (TOP == NULL) // CASO BASE
    TOP = NEWELEM(K)
    RETURN TOP;
ELSE {
    P = INSINCODA(TOP->NEXT, K);
    TOP->NEXT = P;
    RETURN TOP;
}
```

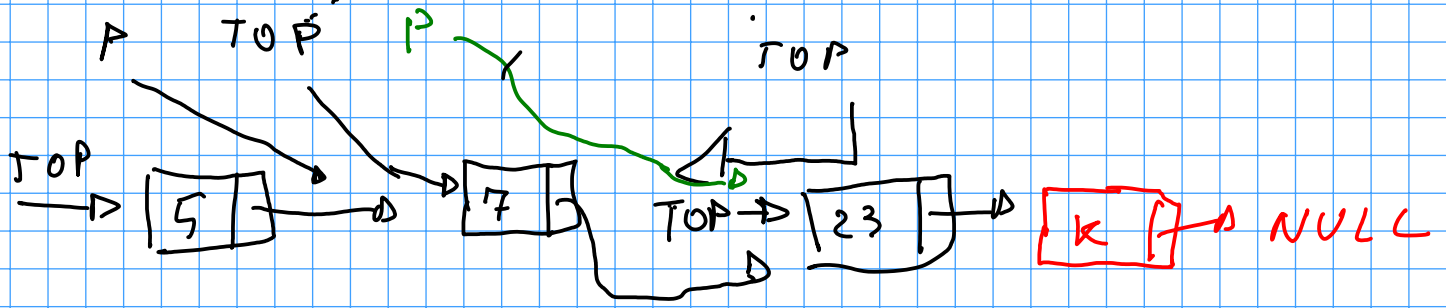
SOLUZIONE DI P(n-1):
"INSERIRE IN CODA NELLA LISTA SENZA LA TESTA, CIOÈ P È IL PUNTATORE ALLA LISTA SENZA TESTA CON K MESSO IN CODA

CASO 1

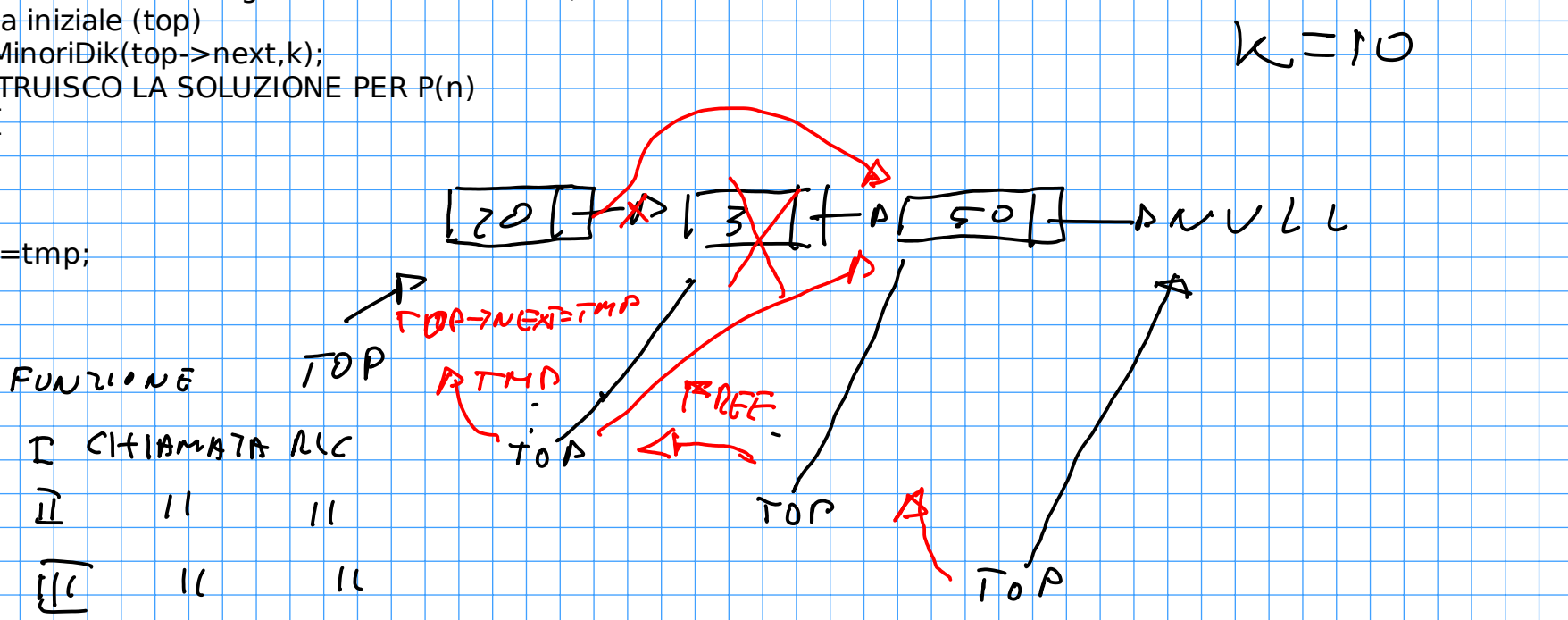


CASO 2

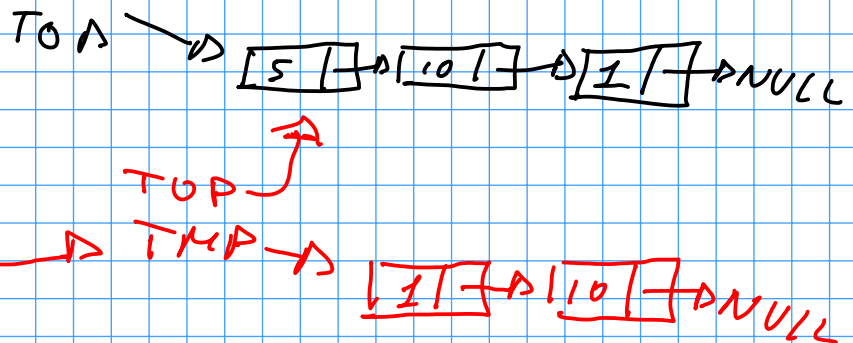
I CHIAMATA RICORSIVA — II CHIAMATA RICORSIVA → III CHIAMATA RICORSIVA



```
if (top!=NULL) {
        //ASSUMMO DI SAPER RISOLVERE IL PROBLEMA P(n-1), cioè
        //P(n)=Eliminare da un lista semplicemente concatenata di dimensione n-1,
        // tutti gli elementi mionori di un dato k
        //tmp mantiene la soluzione, cioè sarà il puntatore alla testa della
        //lista in cui sono stati eliminati gli elementi minori di k,
        //ecceto la testa iniziale (top)
        tmp=cancellaMinoriDik(top->next,k);
        //ADESSO COSTRUISCO LA SOLUZIONE PER P(n)
        if (top->k<k) {
                free(top);
                top=tmp;
                }
        else top->next=tmp;
    }
    return top;
```




```
truct elem *invertiLista(struct elem *top){
    //caso base implicito (top==NULL)
    struct elem *tmp;
    if (top!=NULL) {
        //Assumo che tmp contenga la soluzione per la lista senza la testa,
        //cioè la soluzione del problema P(n-1)=Data una lista semplicemente //
        //concatenata di n-1 interi, voglio invertirla
        tmp=invertiLista(top->next);
        //COSTRUISCO LA SOLUZIONE PER P(n)
    }
}
```



- DEVO INSERIRE TOP IN CODA A TMP, VERIFICANDO ANCHE CHE TMP != NULL