Si considerino delle liste il cui nodo e` costituito dal seguente tipo di dato

```
typedef struct _node {
    int value;
    struct _node * next;
} node;
```

Si definisca la funzione con prototipo

```
node * free_between(node * head, int a, int b);
```

la quale elimina e dealloca ogni elemento della lista con campo value compreso fra **a** e **b**, estremi inclusi.

Se per esempio viene passata la seguente lista come argomento:
head → 1 → 4 → 10 → 5 → 7 → 9

allora, la chiamata free_between(head, 4, 8) restituisce il puntatore alla lista così modificata:
(valore di ritorno) → 1 → 10 → 9

Soluzione iterativa

```
node * free_between(node * head, int a, int b) {
        node * curr, * prev;            /* Puntatori per scorrere lista */

        curr = head;
        prev = NULL;
        while (curr != NULL) {
                if (curr->value >= a && curr->value <= b) {
                        /* Nodo da eliminare */
                        if (prev == NULL) {
                                /* Siamo in testa */
                                curr = curr->next;
                                free(head);
                                head = curr;
                        } else {
                                /* Nodo nel mezzo della lista */
                                prev->next = curr->next;
                                free(curr);
                                curr = prev->next;
                        }
                } else {
                        /* Scorro lista */
                        prev = curr;
                        curr = curr->next;
                }
        }
        return head;
}
```

Si consideri la seguente definizione di tipo:

```
struct string {
   char *seq;
   int   n;
};
```

dove **seq** è il puntatore al primo carattere di una sequenza (già allocata in memoria) di **n** caratteri.

Si definisca la seguente funzione:

```
struct string *f(char *a_Cstring[], int length);
```

che opera su array di stringhe-C ed è tale che:

1. **a_Cstring[]** è un array di **length** elementi, ciascuno dei quali è una stringa C (sequenza di char terminata da zero);
2. la funzione restituisce un array **a** di **length** elementi di tipo **struct string**;
3. l'elemento alla posizione **i** dell'array **a** è così fatto:
   a[i].seq punta alla sequenza di caratteri che sono **una copia** della stringa **a_Cstring[i]**, priva del carattere terminatore;
   a[i].n è la dimensione in caratteri della sequenza.

Ad esempio se:

```
a_Cstring -> { "Hello", " ", "World", "!" }
```

allora la chiamata **f(a_Cstring, 4)** restituisce un array di **struct string** così composto:

```
a -> { {"Hello",5}, {" ",1}, {"World",5}, {"!",1} }
```

```
struct string *f(char *a_Cstring[], int length){

    // array restituito come risultato
    struct string *a = calloc(length, sizeof(struct string));
    // puntatore usato per scorrere per indirizzo gli elementi dell'array risultato
    struct string *e = a;
    // idice usato per scorrere gli elementi dell'array a_Cstring
    int i;

    // ciclo che itera sugli elementi dell'array a_Cstring
    for (i=0 ; i<length; i++, e++){
        int l;     // lunghezza stringa corrente in a_Cstring
        char *end; // puntatore alla posizione che segue l'ultimo elemento
                   // dell'array a_Cstring (fine array)
        char *str = a_Cstring[i];
        char *ptr;

        l = str != NULL ? strlen(str) : 0;
        end = str+l;

        e->seq = ptr = malloc(l);

        while (str<end)
            *ptr++ = *str++;

        e->n = l;
    }
    return a;
}
```