Iniziato	venerdì, 12 settembre 2025, 09:17
Stato	Completato
Terminato	venerdì, 12 settembre 2025, 11:10
Tempo impiegato	1 ora 53 min.
Valutazione	**14,05** su un massimo di 30,00 (**46,82**%)

Domanda **1**

Risposta errata

Punteggio ottenuto 0,00 su 9,00

Completare il codice del seguente programma (NON SI RICHIEDE DI SCRIVERE P-IN, P-OUT e INVARIANTI DI CICLO)

Risposta (Criteri di penalizzazione: 0 %)

Ripristina risposta

```
                    newPtr->next = curr;
                }
                i++;
            }
        } else{
            CharList head = NULL;
            CharList tail = NULL;

            CharList newPtr = (CharList)malloc(sizeof(CharNode));

            newPtr->data = curr->data;
            newPtr->next = NULL;

            if(head == NULL){
                head = tail = newPtr;
            } else{
                tail->next = newPtr;
                tail = newPtr;
            }
        }

        curr = curr->next;

    }

    return 1;
}


/**
 * Invoca la funzione insert sulla lista [Z,A,N,T,E] e sulla stringa "R
 * valore int restituito dalla funzione, poi libera la memoria occupata
 */
int main() {

    char *str = "RON";

    CharList list = (CharList)malloc(sizeof(CharNode));
    list->data = 'Z';
    list->next = (CharList)malloc(sizeof(CharNode));
    list->next->data = 'A';
    list->next->next = (CharList)malloc(sizeof(CharNode));
    list->next->next->data = 'N';
    list->next->next->next = (CharList)malloc(sizeof(CharNode));
    list->next->next->next->data = 'T';
    list->next->next->next->next = (CharList)malloc(sizeof(CharNode));
    list->next->next->next->next->data = 'E';
    list->next->next->next->next->next = NULL;

    printList(list);

    int risultato = insert(&list, str);

    printf("%d\n", risultato);

    printList(list);

    freeList(list);

    return 0;
}
```

	Test	Atteso	Ottenuto	
✖	/** TEST 1: funzione main() **/	.*1.*EXECUTION COMPLETE$	***Errore di esecuzione*** Segmentation fault	✖

Il test è stato arrestato per un errore

Questo è il feeback generale

▾ Visualizza/Nascondi soluzione dell'autore (C)

```c
 * ESEMPI:
 * (1) data *ptr == [A,x,8] ed s == "k$2e"  restituisce 1 e causa *ptr =
 * (2) data ptr == NULL ed s == "Pluto" restituisce 0 e non modifica ptr
 * (3) data *ptr == [] ed s == "Pluto" restituisce 1 e causa *ptr == [P,
 * (4) data *ptr == [P,A,Z,Z,O] ed s == "Pluto" restituisce 1 e causa *p
 * (5) data *ptr == [p,a,z,z,o] ed s == "Pluto" restituisce 1 e causa *p
 * (6) data *ptr == [p,a,z,z,o] ed s == NULL restituisce 1 e non modifi
 */
int insert(CharList *ptr, char const* s) {
    if (ptr == NULL) return 0;
    if (s == NULL) return 1;

    CharList *p = ptr;
    while (*p != NULL && (*p)->data != 'Z') p = &((*p)->next);
    if (*p != NULL) ptr = p;
    for (int i=0; s[i] != 0; ++i) {
        CharList n = malloc(sizeof(CharNode));
        n->data = s[i];
        n->next = *ptr;
        *ptr = n;
        ptr = &(n->next);
    }
    return 1;
}


/**
 * Invoca la funzione insert sulla lista [Z,A,N,T,E] e sulla stringa "RO
 * valore int restituito dalla funzione, poi libera la memoria occupata
 */
int main() {
    CharList list = malloc(sizeof(CharNode));
    list->data = 'Z';
    list->next = malloc(sizeof(CharNode));
    list->next->data = 'A';
    list->next->next = malloc(sizeof(CharNode));
    list->next->next->data = 'N';
    list->next->next->next = malloc(sizeof(CharNode));
    list->next->next->next->data = 'T';
    list->next->next->next->next = malloc(sizeof(CharNode));
    list->next->next->next->next->data = 'E';
    list->next->next->next->next->next = NULL;
    printf("RISULTATO: %d\n", insert(&list, "RON"));
    while (list != NULL) {
        printf("%c",list->data);
        CharList tmp = list;
        list = list->next;
        free(tmp);
    }
    return 0;
}

```

Risposta errata

Punteggio di questo invio: 0,00/9,00.

Domanda **2**

Parzialmente corretta

Punteggio ottenuto 3,55 su 9,00

Dato un albero binario definito da:

```
typedef struct treeNode IntTreeNode, *IntTree;

struct treeNode {
  IntTree left;
  int data;
  IntTree right;
};
```

e la specifica di funzione:

```
/**
* Dato un albero binario tree, e un numero intero taglio, restituisce la somma dei valori di tutti i nodi che soddisfano una
* delle seguenti condizioni:
* 1) sono nodi interni o foglie che si trovano ad un livello pari >= taglio.
* 2) sono foglie il cui livello < taglio. In questo secondo caso al valore del nodo viene sommata
*    la costante 100 (NB: non importa se il livello è pari o dispari.)
* Si assuma taglio>0, e si assuma che il livello della radice sia 0.
*
* ESEMPI:
*
* 1) per tree== NULL, indipendentemente da taglio restituisce 0;
* 2) dato l'albero:
*      1
*     / \
*    2   3
*   / \   \
*  4   5   6
*    /     \
*   7       8
*          / \
*         9   10
* per taglio = 2  restituisce 34 (4 + 5 + 6 + 9 + 10): 4, 5, 6 sono a liv. 2,
* 9 e 10 sono a livello 4. Non ci sono foglie a livello<2.
*
* 3) dato l'albero:
*      1
*     / \
*    2   3
*   / \   \
*  4   5   6
*    /     \
*   7       8
*          / \
*         9   10
* per taglio = 4 restituisce 230  (9 + 10 + 7+100 + 4+100): 9 e 10 sono a liv. 4,
* 7 e 4 sono foglie a un livello minore di taglio quindi per ciascuna si somma 100 al
* valore della foglia.
**/
int sommaTaglio (IntTree tree, int taglio);
```

realizzarne una implementazione RICORSIVA.

Risposta (Criteri di penalizzazione: 0 %)

Ripristina risposta

```
1 int sommaTaglio (IntTree tree, int taglio){
2     if(tree == NULL || taglio < 0) return 0;
3
```

```
4       int count = 0, sum = 0;
5
6       if(count < taglio){
7           if(tree->left == NULL && tree->right == NULL){
8               sum += tree->data + 100;
9
10              return sum;
11          }
12      } else{
13          if(count % 2 == 0){ // pari
14              return tree->data + sommaTaglio(tree->left, taglio) + somma
15          }
16
17      }
18
19      return sum + sommaTaglio(tree->left, taglio) + sommaTaglio(tree->ri
20
21  }
22
```

	Test	Atteso	Ottenuto	
✔	/** ESEMPIO 1 delle specifiche **/	test passed!	test passed!	✔
✖	/** ESEMPIO 2 delle specifiche **/	test passed!	test failed!	✖
✖	/** ESEMPIO 3 delle specifiche **/	test passed!	test failed!	✖
✖	/** ESEMPIO 4: albero delle specifiche con taglio = 1 **/	test passed!	test failed!	✖
✔	/** ESEMPIO 5: albero delle specifiche con taglio = 8 **/	test passed!	test passed!	✔
✔	/** ESEMPIO 6: albero con radice 1 e due figli 8,9 e taglio = 2 **/	test passed!	test passed!	✔
✖	/** ESEMPIO 7: albero con radice 1 e due figli 8,9 e taglio = 1 **/	test passed!	test failed!	✖

	Test	Atteso	Ottenuto	
✔	/** ESEMPIO 8: albero con radice 1 e taglio = 1 **/	test passed!	test passed!	✔
✖	/** ESEMPIO 9: albero degenere 1,2,3,4 e taglio = 3 **/	test passed!	test failed!	✖

Visualizza differenze

Questo è il feeback generale

Parzialmente corretta

Punteggio di questo invio: 3,55/9,00.

Domanda **3**

Risposta corretta

Punteggio ottenuto 6,00 su 6,00

Date le seguenti definizioni

```
typedef struct dLN  *Lista;
struct dLN {
char *name;
Lista next;
        Lista prev;
}

struct mistero {
        int *pn;
Lista gruppi;
} X, a[5], *PTRX;
```

indicare, scegliendo una voce dal menu, quali delle seguenti espressioni sono staticamente (ovvero per il compilatore) corrette e quali sono errate.

1. printf("%d", a[2].pn) è CORRETTA ✔
2. X->pn = PTRX.pn è ERRATA ✔
3. PTRX->pn = X.pn++ è CORRETTA ✔
4. a[1].gruppi.prev = NULL è ERRATA ✔

Risposta corretta.

La risposta corretta è:

Date le seguenti definizioni

```
typedef struct dLN  *Lista;
struct dLN {
char *name;
Lista next;
        Lista prev;
}

struct mistero {
        int *pn;
Lista gruppi;
} X, a[5], *PTRX;
```

indicare, scegliendo una voce dal menu, quali delle seguenti espressioni sono staticamente (ovvero per il compilatore) corrette e quali sono errate.

1. printf(“%d”, a[2].pn) è [CORRETTA]
2. X->pn = PTRX.pn è [ERRATA]
3. PTRX->pn = X.pn++ è [CORRETTA]
4. a[1].gruppi.prev = NULL è [ERRATA]

Domanda **4**

Parzialmente corretta

Punteggio ottenuto 4,50 su 6,00

Date le seguenti definizioni, completare le successive frasi scegliendo l'opzione giusta dal menu:

```
typedef struct dLN  *Lista;
struct dLN {
   char *name;
   Lista next;
         Lista prev;
}

struct mistero {
         int *pn;
   Lista gruppi;
} X, a[5], *PTRX;
```

Completare:

1. l'assegnamento PTRX = a+2 è eseguibile [mai] ✖
2. X è una variabile allocata [staticamente] ✔ che [può talvolta contenere] ✔ una lista di gruppi vuota
3. lassegnamento X.gruppi.next = X.gruppi.prev [restituisce un errore di tipi] ✔

Risposta parzialmente esatta.

Hai selezionato correttamente 3.
La risposta corretta è:
Date le seguenti definizioni, completare le successive frasi scegliendo l'opzione giusta dal menu:

```
typedef struct dLN  *Lista;
struct dLN {
   char *name;
   Lista next;
         Lista prev;
}

struct mistero {
         int *pn;
   Lista gruppi;
} X, a[5], *PTRX;
```

Completare:

1. l'assegnamento PTRX = a+2 è eseguibile [sempre]
2. X è una variabile allocata [staticamente] che [può talvolta contenere] una lista di gruppi vuota
3. lassegnamento X.gruppi.next = X.gruppi.prev [restituisce un errore di tipi]