UNIVERSITÀ DI TORINO Piattaforma Esami

Programmazione II - dal 2024

Started on	Thursday, 6 June 2024, 9:06 AM
State	Finished
Completed on	Thursday, 6 June 2024, 11:06 AM
Time taken	1 hour 59 mins
Grade	**18.58** out of 30.00 (**61.94**%)

Question **1**

Incorrect

Mark 0.00 out of 4.00

Flag question

Realizzare un'implementazione ITERATIVA della seguente specifica di funzione:

/** @brief Verifica se una stringa è palindroma.
*   P-IN(s,first,last): s è una stringa, first e last sono indici validi di caratteri in s
*   P-OUT(s,first,last,result): result è il valore di verità di "la sequenza di caratteri contenuti in s[first...last] è palindroma"
*/
_Bool isPalindrome(const char *s, int first, int last);

Answer: (penalty regime: 0 %)

Reset answer

```c
/** @brief Verifica se una stringa è palindroma.
*    P-IN(s,first,last): s è una stringa, first e last sono in
*    P-OUT(s,first,last,result): result è il valore di verità
*/
_Bool isPalindrome(const char *s, int first, int last){
    // caso base indici first e last sono uguali
    if(first >= last){
        return 1;
    }
    // caso generale : controllo che s[first] == s[last]
    if(s[first]==s[last]){
        return 1 && isPalindrome(s,first+1,last-1);
    }else{
        return 0;
    }
}
```

	Test	Expected	Got
✓	/** TEST 1: stringa vuota **/	test passed!	test passed!
✓	/** TEST 2: singolo carattere **/	test passed!	test passed!
✓	/** TEST 3: stringa palindroma corta **/	test passed!	test passed!
✓	/** TEST 4: stringa non palindroma corta **/	test passed!	test passed!
✓	/** TEST 5: stringa palindroma lunga **/	test passed!	test passed!
✓	/** TEST 6: stringa non palindroma lunga **/	test passed!	test passed!

Passed all tests! ✓

Incorrect

Marks for this submission: 4.00/4.00.

Comment:
Si doveva usare l'iterazione, NON la ricorsione.

Question **2**

Partially correct

Mark 0.58 out of 7.00

Flag question

Date le dichiarazioni:

typedef struct node IntNode, *IntList;

struct node {
  int data;
  IntList next;
}

e la specifica di funzione:

/** @brief Restituisce la lista alternata dei nodi di *lsPtr1 e *lsPtr2, togliendoli da *lsPtr1 e *lsPtr2,
*   che alla fine conterranno entrambi NULL (non alloca nuova memoria).
*   Ad es. date [1, 5, 9] e [0, 2, 4, 6, 8] restituisce [1, 0, 5, 2, 9, 4, 6, 8].
*/
IntList mixAlternate(IntList *lsPtr1, IntList *lsPtr2);

realizzarne un'implementazione ITERATIVA.

Answer: (penalty regime: 0 %)

Reset answer

```c
/** @brief Restituisce la lista alternata dei nodi di *lsPtr1
*   che alla fine conterranno entrambi NULL (non alloca nuova
*   Ad es. date [1, 5, 9] e [0, 2, 4, 6, 8] restituisce [1, 0,
*/
IntList mixAlternate(IntList *lsPtr1, IntList *lsPtr2){


    // controllo la validita dei nodi
    if(*lsPtr1 == NULL && *lsPtr2 == NULL ) return NULL;
    // creo la lista di appoggio
    IntList new = *lsPtr1;
    new->next = *lsPtr2;


    // una volta creata la lista di appoggio inizio a scorrerla
        // 1 : se entrambi i nodi sono validi allora copio il
        // 2/3 : se uno solo è valido continuero a copiare que
    while((*lsPtr1)->next!=NULL && (*lsPtr2)->next!=NULL){

       // caso entrambe valide
       if(lsPtr1 && lsPtr2){
            new->data=(*lsPtr1)->data; // assegno al nuovo nod
            new->next=*lsPtr2; // e al next il secondo
            new->next->next = NULL;
            printf("%d",new->data);

          *lsPtr1 = (*lsPtr1)->next; // scooro il primo e il se
          *lsPtr2 = (*lsPtr2)->next;

       } else if(lsPtr2==NULL || *lsPtr2==NULL){
           printf("quo");
           new = *lsPtr1;// se è valido solo il primo allora s
           new = new->next;

          *lsPtr1 = (*lsPtr1)->next;

       }else if(lsPtr1 == NULL || *lsPtr1==NULL){
            printf("qua");
            new = *lsPtr2;
            new = new->next;
           *lsPtr2 = (*lsPtr2)->next;
       }

    }

    free(*lsPtr1);
    free(*lsPtr2);
    return new;
}
```

	Test	Expected	Got
✓	/** TEST 1: liste NULL e NULL **/	test passed!	test pas
✗	/** TEST 2: liste [4,3] e NULL **/	test passed!	test fai
✗	/** TEST 3: liste [4,3] e [5,2] **/	test passed!	3test fa
✗	/** TEST 4: liste [1, 5, 9] e [0, 2, 4, 6, 8] **/	test passed!	9test fa
✗	/** TEST 5: liste [0, 2, 4, 6, 8] e [1, 5, 9] **/	test passed!	8test fa

Show differences

Questo è il feeback generale

Partially correct

Marks for this submission: 0.58/7.00.

Question **3**

Correct

Mark 7.00 out of 7.00

Flag question

Dato un albero albero binario definito da:

typedef struct treeNode IntTreeNode, *IntTree;

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

e la specifica di funzione:

/**@brief Trasforma un albero nella sua versione speculare. */
void mirror(IntTree tree);

realizzarne una implementazione RICORSIVA.

Ad esempio la versione speculare di:

```
  2
 / \
1   5
   / \
  4   9
```

è:

```
   2
  / \
  5  1
 / \
9   4
```

Answer: (penalty regime: 0 %)

Reset answer

```c
/** @brief Trasforma un albero nella sua versione speculare. */
void mirror(IntTree tree){
    if(tree){
    // caso base l e r == NULL
    if(tree->left == NULL && tree->right == NULL){
        return;//non faccio nulla
    }

    //altrimenti chiamo la funzione ricorsiva su left e right
    mirror(tree->left);
    mirror(tree->right);
    IntTree tmp = tree->left;
    tree->left = tree->right;
    tree->right = tmp;

    }
}
```

	Test	Expected	Got	
✓	/** TEST 1 **/	test passed!	test passed!	✓
✓	/** TEST 2 **/	test passed!	test passed!	✓
✓	/** TEST 3 **/	test passed!	test passed!	✓
✓	/** TEST 4 **/	test passed!	test passed!	✓

Passed all tests! ✓

Questo è il feeback generale

Correct

Marks for this submission: 7.00/7.00.

Question **4**

Correct

Mark 3.00 out of 3.00

Flag question

Si considerino le seguenti dichiarazioni:

```c
typedef struct _scheda Scheda, *Puntatore;
struct _scheda {
    int numero;
    char *testo;
}
int funzione(Scheda sc, Puntatore pu);
Scheda x;
Scheda *y
Puntatore *z;
```

Quali delle seguenti invocazioni sono staticamente (ovvero per il compilatore) corrette e quali sono invece errate?

funzione(x, NULL); CORRETTA ✓
funzione(**z, y); CORRETTA ✓
funzione(y, *y); ERRATA ✓

Risposta corretta.

Question **5**

Correct

Mark 3.00 out of 3.00

Flag question

Si consideri il codice riportato nel seguito e si risponda alle domande:

```c
int arr[] = {1, 4, -1, 5};

int funz(int *a, int n) {
   if (a == NULL) return -1;
   if (n < 0) return -2;
   int *ptr = a;
   int i = 0;
   int sum = 0;
   while (i < n-1) {
         if (a[i]%2 == 0) { i++; sum = sum +  *(ptr+i); }
         else  { sum = sum +  ptr[i+1]; i++; }
     }
  return sum;
}
```

la funzione può produrre un segmentation fault VERO ✓

se non produce un segmentation fault, la funzione terminerà sempre VERO ✓

La chiamata di funzione funz(arr,4) restituisce: 8 ✓

Risposta corretta.

Question **6**

Partially correct

Mark 5.00 out of 6.00

Flag question

Date le seguenti definizioni

```c
enum tag_paese {UK, ITALIA};

typedef struct _indUK {
    int number;
    char street[N1];
    int floor;
    char pcode[N2];
    char town[N3];
} indUK;

enum tag_strada {VIA, PIAZZA, CORSO};

typedef struct _indITA {
    int civico;
    enum tag_strada strada;
    char  nomestrada[N1];
    char citta[N3];
} indITA;

struct {
    enum tag_paese nazione;
    union {
        indUK address;
        indITA indirizzo;
    } n;
} X, X2, *PTRX;
```

indicare quali delle seguenti espressioni sono staticamente (ovvero per il compilatore) corrette e quali sono errate.

X.n.address.civico = 10; ERRATA ✓
if (nazione == UK) strcpy(X.n.address.street, "Magellan Lane"); CORRETTA ✗
PTRX = &X; CORRETTA ✓
PTRX = &(X->nazione); ERRATA ✓
PTRX->n.indirizzo.nomestrada[0] == 'z'; CORRETTA ✓
PTRX->nomestrada ERRATA ✓

Risposta parzialmente esatta.

You have correctly selected 5.

Finish review

Quiz navigation

1 2 3 4 5 6

Show one page at a time

Finish review