Complementi di Programmazione

Esercitazione 10

- Scrivere dei test nel main per verificare che le funzioni scritte siano corrette.
- Manipolare le pile e le code solo tramite le funzioni del tipo astratto (vedere pila.h e coda.h), non accedere all'implementazione.

Pile

Esercizio 10.1

Implementare la funzione

```
void stampa_sequenza_inversa(void);
```

che legge una sequenza di numeri inseriti da tastiera. Quando in input viene inserito un numero negativo, la sequenza termina, e la funzione deve stampare in output tutta la sequenza inserita fino a quel momento, in ordine inverso.

Esercizio 10.2

Scrivere una funzione:

```
void rimuovi_alcuni(Pila *p, const int *rimuovere, int n)
```

che riceve una pila `p`, e un array `rimuovere` di lunghezza `n`. La funzione deve ripetutamente controllare se il primo elemento di `p` è presente in `rimuovere`, e eventualmente rimuoverlo dalla pila. Questo processo termina al primo elemento che non viene trovato in `rimuovere`.

Esempio:

p= [1,2,3,4,5,6]
rimuovere=[1,2,3]
Risultato:
[4,5,6]

Esercizio 10.3

Scrivere la funzione:

```
bool check_palindrome(const char* s);
```

che, data una stringa in input, restituisce true se la stringa rappresenta un'espressione palindroma di lunghezza pari. Se `s` è vuota, si restituisca true. E' consentito l'utilizzo di `strlen`. La funzione deve accedere agli elementi una sola volta, accedendo ordinatamente dal primo all'ultimo.

Esempio

s: "aa" ret: true
s: "a" ret: false
s: "" ret: true
s: "abba" ret: true
s: "abb" ret: false
s: "145541" ret: true

Esercizio 10.4

Scrivere la funzione:

```
bool check_palindrome_r(const char* s);
```

che effettua lo stesso calcolo di `check_palindrome`. `check_palindrome_r` deve essere ricorsiva, e non utilizzare alcuna pila. E' ancora permesso l'uso di strlen.

Esercizio 10.5

Scrivere la funzione:

```
int* rimuoviMinori(int* arr, int len);
```

che, dati in input un array *arr*, contenente una sequenza di numeri, e la sua lunghezza *len*, restituisce un array costruito nel seguente modo usando una pila:

1) Aggiungere i numeri alla pila uno alla volta
2) Ogni volta che il numero successivo è minore dell'ultimo numero inserito nella pila, rimuovere ciascun elemento della pila che è maggiore di tale numero fermandosi al primo elemento minore o uguale, dunque inserire il prossimo numero nell'array.

Esempio:

arr = {1, 3, 4, 2, 4, 2, 3}

posizione_in_array = 0, elemento_corrente = 1,
elementi_aggiunti = {1}
pila = {1}

posizione_in_array = 1, elemento_corrente = 3
elementi_aggiunti = {3}
pila = {3,1}

posizione_in_array = 2, elemento_corrente = 3
elementi_aggiunti = {4}
pila = {4,3,1}

posizione_in_array = 3, elemento_corrente = 2
elementi_rimossi = {4,3}, elementi_aggiunti = {2}
pila = {2,1}

posizione_in_array = 4, elemento_corrente = 4
elementi_aggiunti = {4}
pila = {4,2,1}

posizione_in_array = 5, elemento_corrente = 2
elementi_rimossi = {4}, elementi_aggiunti = {2}
pila = {2,2,1}

posizione_in_array = 6, elemento_corrente = 3
elementi_aggiunti = {3}
pila = {3,2,2,1}

array_risultante = {3,2,2,1}

Code

Esercizio 10.6

Implementare la funzione

```
void stampa_sequenza(void);
```

che legge una sequenza di numeri inseriti da tastiera. Quando in input viene inserito un numero negativo, la sequenza termina, e la funzione deve stampare in output tutta la sequenza inserita fino a quel momento.

Esercizio 10.7

Implementare la funzione

```
int elemento_iesimo(Coda *c, int pos)
```

Che data una coda `c` in input, rimuova i primi `pos` elementi e restituisca il valore in posizione `pos`. Se la posizione non è nella coda, si restituisca il valore -1.

Esercizio 10.8

Implementare la funzione

```
void avoid_stampa(Coda *c, int elem)
```

Che data una coda `c` in input, stampi tutti gli elementi della coda tranne quelli uguali ad `elem`. Dopo l'esecuzione della funzione, la coda `c` deve ancora contenere gli elementi di partenza.

Esercizio 10.9

Implementare la funzione:

```
Coda * coda_circolare(Coda *c, int n);
```

che data una coda `c` in input e un numero di elementi da stampare `n`, restituisce una nuova coda di `n` elementi, ottenuta dalla coda `c` nel seguente modo:

1) Se n è minore o uguale al numero di elementi contenuti in c, la coda risultante conterrà i primi n elementi di c
2) Se n è maggiore del numero di elementi in c, la coda conterrà gli elementi di c, in sequenza, ricominciando dal primo, ogni qualvolta la coda viene "esaurita", finché non si arriva ad n.

Esempi:

c = {0,1,2,3}, n = 2
Output: "0,1"

c = {0,1,2,3}, n = 4
Output: "0,1,2,3"

c = {0,1,2,3}, n = 6
Output: "0,1,2,3,0,1"

c = {0,1,2,3}, n = 10
Output: "0,1,2,3,0,1,2,3,0,1"

Esercizio 10.10

Implementare la funzione

```
Coda * elementi_pari(Coda *c)
```

Che data una coda c1 in input, restituisca una nuova coda con solo gli elementi di c in posizione pari.