UNIVERSITÀ DEGLI STUDI DI BERGAMO
Dipartimento di Ingegneria Gestionale, dell'Informazione e della Produzione

Programmazione Avanzata

Durata: 4h
Data: 21 giugno 2024
Punteggio Massimo: 26 PUNTI.

Chiama i tuoi progetti con MATRICOLA_ESX con X il numero dell'esercizio. Per esempio, 2574188_ES1 conterrà la soluzione dell'esercizio 1 dello studente con matricola 2574188.

Consegna: crea uno zip unico e carica il file su Moodle [cartella `` Programmazione Avanzata 2024-06-21''].

1. Funzione in C (4 punti)

Scrivi una funzione **sort** che riceve un array di interi e **restituisce** una copia dell'array ordinato.

(a) Scrivi tre versioni:
- Una iterativa non ricorsiva
- Una ricorsiva senza tail call
- Una ricorsiva con tail call.

(b) Scrivi anche un main di esempio in cui chiami le funzioni con almeno gli array di cui sopra.

Non usare alcuna variabile globale. Cerca di tenere il più semplice possibile la segnatura delle funzioni ricorsive, ma se non riesci fai una funzione con segnatura semplice che chiami quella ricorsiva.

2. Record di Attivazione (4 punti)

Si consideri la seguente funzione:

```
int pow_n(int a, int ex) {
      if(ex==0)  return 1 ;
      if(ex==1) return a;
      return a*(pow_n(a,(ex-1)));
} ;
```

Si mostrino i record di attivazione nello stack quando chiamata con a=5, ex=3: si crei un file .txt composto come segue (si aggiungano righe al file se necessario)

Label	Indirizzo	Contenuto	Note

3. Tipi opachi (3 punti)

Definisci il tipo opaco List che rappresenta una lista di interi. Definisci le seguenti operazioni:

costruttore: crea una lista ed inserisce l'insieme di interi contenuti in un array passato come argomento (facendone una copia).
concatena: concatena la lista con un'altra lista e restituisce una nuova lista contenente la loro concatenazione.
print: stampa la lista.
cancella: distrugge la lista e libera la memoria

Implementa la lista mediante una lista puntata.

Fai un esempio in cui:

- Costruisci la lista con gli elementi {1,2,3} e lo stampi.
- Calcola la concatenazione delle liste contenenti gli elementi {1,4,5,7} e {8,4,3,1}.
- Cancelli tutte le liste.

4. Inizializer list in C++ (1.5 punti)

Definire una classe Partita che rappresenta una partita di EURO 2024. La classe partita ha due attributi di classe Nazionale che rappresentano le nazionali di calcio che giocano la partita. Definire una classe Nazionale che ha come attributo il nome del relativo paese. Sviluppare tre implementazioni diverse della classe Partita in cui i suoi attributi sono rispettivamente (a) due puntatori a una porzione di memoria nello stack, (b) due puntatori a una porzione di memoria nell'Heap, (c) due riferimenti a una porzione di memoria nell'Heap. Per ciascuno di questi tre casi discutere se (a) è possibile utilizzare l'inizializer list e (b) è possibile inizializzarlo all'interno del costruttore.

5. Smart pointers in C++ (1.5 punti)

Considerare la classe Partita dell'esercizio 4. Crea una Partita utilizzando un raw pointer, uno smart pointer, e uno shared smart pointer. Alla distruzione della Partita devono essere distrutte anche la relativa nazionali. Illustra e spiega le differenze relative all'utilizzo dei vari tipi di puntatori creando un metodo main e commentando opportunamente il codice.

6. Java Generics (4 punti)

Implementa la struttura dati "List" dell'esercizio 3 in Java, che però sia generica in modo che si possa memorizzare ogni tipo che estenda Object. Come tipo utilizza una lista. Crea un "main" che illustri l'utilizzo di List creando una lista di interi ed una lista di stringhe.

7. Visitor pattern (4 punti)

Si vuole sviluppare una applicazione per gestire l'accesso allo stadio degli europei di calcio (EURO 2024). Uno spettatore ha un nome ed un cognome. Ci sono tre tipologie di spettatori: VIP, tifosi, e giornalisti. Ogni giornalista ha una testata giornalistica a cui appartiene. A differenza dei giornalisti e ai VIP (che hanno dei settori riservati), che hanno accesso alla tribuna VIP, i tifosi hanno accesso a una tribuna specifica. È possibile settare il posto assegnato ad un tifoso mediante un opportuno metodo setPosto(int tribuna, int posto).

(a) Si vuole utilizzare il pattern Visitor per
    1) Controllare se l'utente ha accesso alla tribuna VIP.
    2) Stampare nome e cognome dello spettatore.
(b) Deve essere possibile ordinare i tifosi in ordine crescente in base al loro posto ed alla loro tribuna. Utilizzare l'interfaccia Comparable e mostrare un esempio nel quale viene creata una lista di tifosi e vengono ordinati in base al loro posto e tribuna.
(c) Creare una Lista di elementi di tipo Tifoso. Discuti se è possibile assegnare questa lista ad un Lista di elementi di tipo Spettatore e presenta le relative motivazioni.

8. Programmazione funzionale (4 punti)

(a) Definire in Scala la funzione *raddoppia*: Int=>Int che restituisce il doppio di un valore intero passato come parametro.
(b) Definire la funzione *somma* che data una funzione *f*: Int=>Int restituisce una funzione che prende due parametri interi (a e b, tali che a<b) e restituisce la somma di f(x) per ogni x compreso tra a e b (a e b inclusi).
(c) Definire la funzione *raddoppiaESomma* che applica la funzione *raddoppia* e la funzione *somma*
(d) Utilizzare la funzione *raddoppiaESomma* passando come parametri a=4 e b=6.
(e) Definire una lista di interi contenente i valori 1,2,3,4 per il parametro "a". Applicare la funzione *raddoppiaESomma* per ogni elemento della lista.
(f) Utilizzare map-reduce per sommare i valori al punto (e).