PROGRAMMAZIONE II (A,B) - a.a. 2013-14
Primo Appello — 12 giugno 2014

Esercizio 1. Si consideri il tipo di dati astratto modificabile `BoundedObjectSequence`, i cui elementi sono collezioni finite di oggetti rappresentati come sequenze. Supponiamo che `BoundedObjectSequence` fornisca le operazioni definite dalla seguente interfaccia:

```
public interface BoundedObjectSequence <T> {
  boolean insert(T item) throws IllegalArgumentException
  // Se la sequenza non e' piena inserisce l'oggetto item all'inizio
  // della sequenza e restituisce true, altrimenti restituisce false.
  // Lancia l'eccezione se item e' null.
  T get(int i)
  // Restituisce l'i-esimo elemento della sequenza, se esiste, null altrimenti
  void remove(T item) throws ElementNotFoundException
  // Rimuove la prima occorrenza dell'elemento item dalla sequenza.
  // Lancia una ElementNotFoundException se non esiste.
  void removeAll(T item)
  // Rimuove tutte le occorrenze dell'elemento item dalla sequenza
  boolean contains(T item)
  // Restituisce true se la sequenza contiene l'elemento item
  boolean isEmpty()
  // Restituisce true se la sequenza e' vuota
  boolean isFull()
  // Restituisce true se la sequenza e' piena
```

1. Completare la specifica del tipo di dati astratto `BoundedObjectSequence`, descrivendo l'overview e la specifica completa dei costruttori, dei metodi e delle eccezioni. Il numero massimo di elementi che può contenere una `BoundedObjectSequence` è determinato al momento della creazione, e non è modificabile.

2. Definire una struttura di implementazione per il TdA `BoundedObjectSequence` basata su di un array che contiene gli oggetti della sequenza. Definire l'invariante di rappresentazone e la funzione di astrazione.

3. Scrivere una implementazione del metodo `removeAll`. Dimostrare che l'implementazione proposta preserva l'invariante di rappresentazione.

4. Aggiungere al TdA la specifica del metodo `T[ ] toArray()`, che deve restituire un array contenente tutti gli elementi della sequenza. Discutere come si può implementare il metodo senza rompere l'incapsulamento del TdA.

5. Definire la specifica del TdA `ObjectSequence`, che estende la specifica `BoundedObjectSequence` realizzando sequenze di oggetti di dimensione arbitraria. Discutere se la specifica così ottenuta rispetta il principio di sostituzione.

Esercizio 2. Si consideri il seguente programma scritto in sintassi C-like:

```
void add (int arr[], int size, int i, int v} {
  if (i>= 0 && i < size)   arr[i] = arr[i]+v;
      else failwith "wrong array index";
}

double getAverage(int arr[], int size){
  int i;
  double avg;
  int sum;
  for (i = 0; i < size; ++i){
    sum += arr[i];
  }
  avg = sum / size;
  return avg;
}

int main (){
   int a[5] = {10, 20, 30, 40, 50};
   double avg;
   avg = getAverage(a, 5);
   printf( "Prima media = %f ", avg );
   add(a, 5, 3, 60);
   avg = getAverage(a, 5);
   printf( "Seconda media = %f ", avg );
   return 0;
}
```

Mostrare cosa stampa il programma e come evolve la struttura dello stack a run-time nei due casi seguenti:

1. Passaggio di array come parametri per riferimento;
2. Passaggio di array come parametri per valore.

Si ignorino i record di attivazione della funzione `printf`.

Esercizio 3. Estendere il linguaggio funzionale didattico (senza funzioni) con un meccanismo per la valutazione *lazy* di espressioni.

Si aggiungano al linguaggio le primitive `lazy(E)` e `lazyForce(E)` con il seguente significato. L'espressione `lazy(E)` congela l'espressione `E` in modo da posticiparne la valutazione. La valutazione di `lazyForce(E1)` restituisce un valore solo se `E1` è un'espressione lazy, altrimenti fallisce. In particolare la valutazione di `lazyForce(lazy(E))` produce lo stesso risultato della valutazione di `E`. Naturalmente un'espressione lazy può contenere identificatori, che vanno risolti con scoping statico.

1. Estendere la sintassi astratta del linguaggio didattico funzionale in modo da includere il meccanismo di valutazione lazy appena descritto
2. Definire le regole di semantica operazionale della valutazione dei nuovi costrutti e derivare il relativo codice OCaml dell'interprete.