Pumping lemma per i linguaggi regolari (enunciato, dimostrazione, applicazione) **(4 punti).**

Teorema dell’alt **(4 punti).**

Definire il DFA che accetta le stringhe in {0,1}\* che contengono almeno un “1” e hanno un numero pari di “0” dopo l’ultimo “1” **(4 punti).**

Definire il NFA che riconosce il linguaggio rappresentato dall’espressione regolare a(a+b+c)\*(a+c)c, trasformarlo nel corrispondente DFA e scrivere la relativa grammatica regolare **(4 punti).**

Definire il NFA che riconosce il linguaggio rappresentato dall’espressione regolare a(a+ba)\*a, trasformarlo nel corrispondente DFA e scrivere la relativa grammatica regolare **(4 punti).**

Definire il NFA che riconosce l’espressione (a+b)\*ba\*(a+b), trasformarlo nel corrispondente DFA e scrivere la relativa grammatica regolare **(4 punti).**

Definire il DFA che riconosce le stringhe formate da un numero pari di “0” e di “1” **(4 punti).**

Definire il NFA che riconosce l’espressione (a+b)\*b(a+b)+, trasformarlo nel corrispondente DFA e scrivere la relativa grammatica regolare **(4 punti).**

Definire il NFA che riconosce l’espressione (a+b)\*b(a+b)(a+b), trasformarlo nel corrispondente DFA e scrivere la relativa grammatica regolare **(4 punti).**

Definire il NFA che riconosce l’espressione (a+b)\*ba\*(a+b)\*, trasformarlo nel corrispondente DFA e scrivere la relativa grammatica regolare. **(4 punti)**

Definire il NFA che riconosce le stringhe date da (a+b)\*bb(a+b), trasformarlo nel corrispondente DFA e scrivere la relativa grammatica regolare **(4 punti).**

Possibili implementazioni del comando WHILE B DO C, con vantaggi e svantaggi **(4 punti).**

Scrivere la grammatica che genera il linguaggio delle parole in {a,b}+, con il numero di a diverso dal numero di b **(4 punti).**

Scrivere la grammatica che genera il linguaggio {w {a,b,c}+ | w=anbncn } **(4 punti).**

Scrivere la grammatica che genera il linguaggio {w {a,b,c}+ | w=anbnc2n } **(4 punti).**

Scrivere la grammatica che genera il linguaggio delle parole in {a, b, c}\* senza doppie **(2 punti).**

Definire la grammatica di tipo 2 che genera il linguaggio delle parole formate da un numero uguale di “0” e “1” (**4 punti**).

Dimostrare che il linguaggio rappresentato da una espressione regolare può essere riconosciuto da un automa a stati finiti (**4 punti**).

Dimostrare che per ogni DFA M possiamo costruire una grammatica regolare G tale che L[M]=L[G] (**4 punti**).

Scrivere la funzione somma come una tail recursion **(4 punti).**

Dire cosa viene stampato (e mostrare la pila di sistema) dal seguente frammento di codice scritto in un linguaggio che usa scope statico e passaggio per valore **(4 punti).**

{ **int** x=2;

**void** A (**value int** y) { x++;

write(y);

y++ ; }

A(x+1)

write (A(x)); }

Dire cosa viene stampato (e mostrare la pila di sistema) dal seguente frammento di codice scritto in un linguaggio che usa scope dinamico e passaggio per valore **(4 punti).**

{ **int** x=2;

**void** A (**val int** y) { x++;

write(y);

y++ ; }

A(x)

write (x); }

Dire cosa viene stampato (e mostrare la pila di sistema) dal seguente frammento di codice scritto in un linguaggio che usa scope statico e passaggio per nome **(4 punti).**

{ **int** x=2;

**void** A (**name int** y) { x++;

write(y);

x++ ; }

A(x+1)

write (x); }

Dire cosa viene stampato (e mostrare la pila di sistema) dal seguente frammento di codice scritto in un linguaggio che usa scope dinamico e passaggio per valore/risultato **(4 punti).**

{ **int** x=2;

**void** A (**val/res int** y) { x++;

write(y);

y=y+2; }

A(x)

write (x); }

Dire cosa viene stampato (e mostrare la pila di sistema) dal seguente frammento di codice scritto in un linguaggio che usa scope dinamico e passaggio per nome **(4 punti).**

{ **int** x=2;

**void** A (**name** y) { x++;

write(y);

return y+x; }

A(x)

write (x); }

Dire cosa viene stampato (e mostrare la pila di sistema) dal seguente frammento di codice scritto in un linguaggio che usa scope statico e passaggio per nome **(4 punti).**

{ **int** x=2;

**void** A (**name int** y) { x++;

write(y);

write(++x + y) ; }

A(x+1)

write (x); }

Scrivere la funzione prodotto come una tail recursion **(4 punti).**

Si consideri il frammento di codice seguente, con modello delle variabili a riferimento e che adotta una tecnica a scelta fra locks-and-keys e tombstone (C è una classe qualsiasi); fornire la situazione della memoria alla fine del codice e dire se e quale parte può essere restituita alla lista libera **(4 punti).**

C a = new C();

C b = new C();

C c = a;

b=a;

Dato il seguente frammento di programma, si descriva graficamente la pila dei record di attivazione, limitatamente ai puntatori di catena statica e dinamica, dopo le chiamate P1, P2, P3, P4, P2 (le chiamate sono tutte attive) **(4 punti).**

**void** P1 {

**void** P2 { *corpo-di-P2* }

**void** P3 {

**void** P4 { *corpo-di-P4* }

*corpo-di-P3*

}

*corpo-di-P1* }

Dire cosa viene stampato (e mostrare la pila di sistema) dal seguente frammento di codice scritto in un linguaggio che usa scope dinamico e passaggio per valore/risultato **(4 punti).**

{ **int** x=2;

**void** A (**val/res int** y) { int x=3 ;

x++;

write(y);

y++ ; }

A(x)

write (x); }

Forma di un array e sua allocazione in memoria **(4 punti).**

Implementazione dello scope statico tramite catena statica **(4 punti).**

Implementazioni dell’ereditarietà singola, con vantaggi e svantaggi **(4 punti).**

Scrivere la grammatica che genera il linguaggio {w=anb2ncn , con n maggiore o uguale a 1} **(4 punti).**

Descrivere graficamente la situazione delle vtable e della memoria al momento della chiamata del metodo f, rispetto al seguente codice: **(4 punti)**

|  |  |
| --- | --- |
| class A { int a;  char c;  void g(){…}  void f(){…} } | class B extending A {int a;  int b;  void h(){…}  void f(){…} } |
| B pb = new B()  A pa = new A()  A aa = pb  aa.f() |  |

Equivalenza e compatibilità fra tipi **(4 punti).**

Implementazione compilativa pura, interpretativa pura, confronto fra le tecniche.

La macchina intermedia **(4 punti).**

Definizione di grammatica ambigua ed esempio **(4 punti).**

Dire cosa viene stampato (e mostrare la pila di sistema) dal seguente frammento di codice scritto in un linguaggio che usa scope dinamico e passaggio per valore/risultato **(4 punti).**

{ **int** x=2;

**void** A (**val/res int** y) { int x=3 ;

x++;

write(y);

y++ ; }

A(x);

write (x);

A(x);

write (x); }

Dire cosa viene stampato dal seguente frammento di codice scritto in un linguaggio che usa scope statico e passaggio per valore **(4 punti).**

{ **int** x=0;

**int** A (**val int** y) { **int** x=2;

y=y+1;

**return** B(y)+x ; }

**int** B (**val int** y) {

**int** C (**val** **int** y) { **int** x=3;

**return** A(y)+x+y ;}

**if** (y==1) **return** C(x)+y ; **else return** x+y ;}

write (A(x)); }

Fornire la definizione di grammatica libera da contesto, derivazione diretta, derivazione, linguaggio generato, albero di derivazione; per ogni definizione fornire un esempio adeguato. **(4 punti)**

Espressioni: sintassi, semantica, valutazione delle espressioni **(4 punti).**

Cosa contribuisce a determinare l’ambiente di valutazione di un linguaggio strutturato a blocchi? Per ogni elemento fornire definizioni ed esempi. **(4 punti)**

Deep binding: definizione, esempi e implementazione. **(4 punti)**

Funzioni come parametro (**4 punti**).

Definire il concetto di polimorfismo di un sistema di tipi; di overloading; di polimorfismo universale parametrico; di polimorfismo universale di sottotipo (**4 punti**).

Implementazione dello scope dinamico: A-list, CRT, CRT con pila nascosta (**4 punti**)

Dato il seguente frammento di programma, si descriva graficamente la pila dei record di attivazione e la catena statica (implementata con il display), dopo le chiamate

P1, P2, P3, P4, P2, P3, P5 **(4 punti).**

**void** P1 {

**void** P2 { *corpo-di-P2* }

**void** P3 {

**void** P4 { *corpo-di-P4* }

**void** P5 { *corpo-di-P5* }

*corpo-di-P3*

}

*corpo-di-P1* }

Scrivere la grammatica che genera il linguaggio delle parole in {a,b}+, con il numero di a doppio del numero di b **(4 punti).**

Dire cosa viene stampato dal seguente frammento di codice scritto in un linguaggio che usa scope statico e passaggio come indicato **(4 punti).**

{ **int** x=0;

**int** A (**val int** y) { **int** x=2;

y=y+1;

**return** B(y)+x ; }

**int** B (**ref int** y) {

**int** C (**val** **int** y) { **int** x=3;

**return** A(y)+x+y ;}

**if** (y==1) **return** C(x)+y ; **else return** x+y ;}

write (A(x)); }

Dimostrare l’equivalenza fra la classe dei linguaggi riconosciuti dai DFA e la classe dei linguaggi riconosciuti dai NFA. **(4 punti)**

Dimostrare che ogni linguaggio accettato da un NFA è accettato da un DFA **(4 punti)**

Definiti i linguaggi L1={aa, ab, bc} e L2={22, 31}, descrivere i linguaggi L1\*, L1L2, L1L2\*, L1∪L2. Nei casi con cardinalità infinita, riportare solo alcune delle parole che li compongono (**4 punti**)

Scrivere un’espressione regolare per le stringhe binarie di lunghezza dispari (**4 punti**)

Dire cosa viene stampato (e mostrare la pila di sistema) dal seguente frammento di codice scritto in un linguaggio che usa scope statico e passaggio per nome **(4 punti)**

{**int** x=2;

**void** A (**name int** y) {int x=7;

write(y+x);

x++ ; }

A(x+1)

write (x); }

Calcolo degli indirizzi di un elemento generico di un array multidimensionale **(4 punti)**

Catena dinamica, A-list, CRT, CRT con pila nascosta; confronto fra le tecniche (**4 punti**)

Definire il NFA che riconosce il linguaggio rappresentato dall’espressione regolare a(a+b+c)\*(a+c)c. Produrre il relativo DFA. Scrivere la relativa grammatica regolare **(4 punti).**

Scrivere la grammatica che genera il linguaggio {w=anb2nc3n , con n positivo} **(4 punti).**

Dire cosa viene stampato (e mostrare la pila di sistema) dal seguente frammento di codice scritto in un linguaggio che usa scope dinamico e passaggio per valore/risultato **(4 punti).**

{ **int** x=2;

**void** A (**val/res int** y) { x++;

write(y);

y=y+2; }

A(x)

A(x+1)

write (x); }

Implementazioni dell’ereditarietà singola, con vantaggi e svantaggi **(4 punti).**

Semantica operazionale dei linguaggi operativi **(4 punti).**

Deep binding: definizione, esempi e implementazione. **(4 punti)**