

طراحی کامپایلرها

نیم سال دوم ۹۹-۹۸

نام و نام خانوادگی: حسن ذاکر، علیرضا دقیق، سپهر فعلی



دانشکده مهندسی کامپیوتر

موعد تحویل: ۹۹/۰۹/۲۲

SDT, Scope, Type Checking

پاسخ تمرین سری سوم

ما ۳ نفر همه سوالات را با هم حل کردیم
مسئله ۱. اسکوپ

پاسخ.

۱. Static Scope

خروجی:

13 0 2

13 14 2

۲. Dynamic Scope

محتویات Symbol Table تا انتهای تابع B و قبل از بستن آن آمده است.

x	10
y	10
z	12
z	5
x	6
y	7
y	0
x	6
z	2

خروجی نیز:

6 0 2

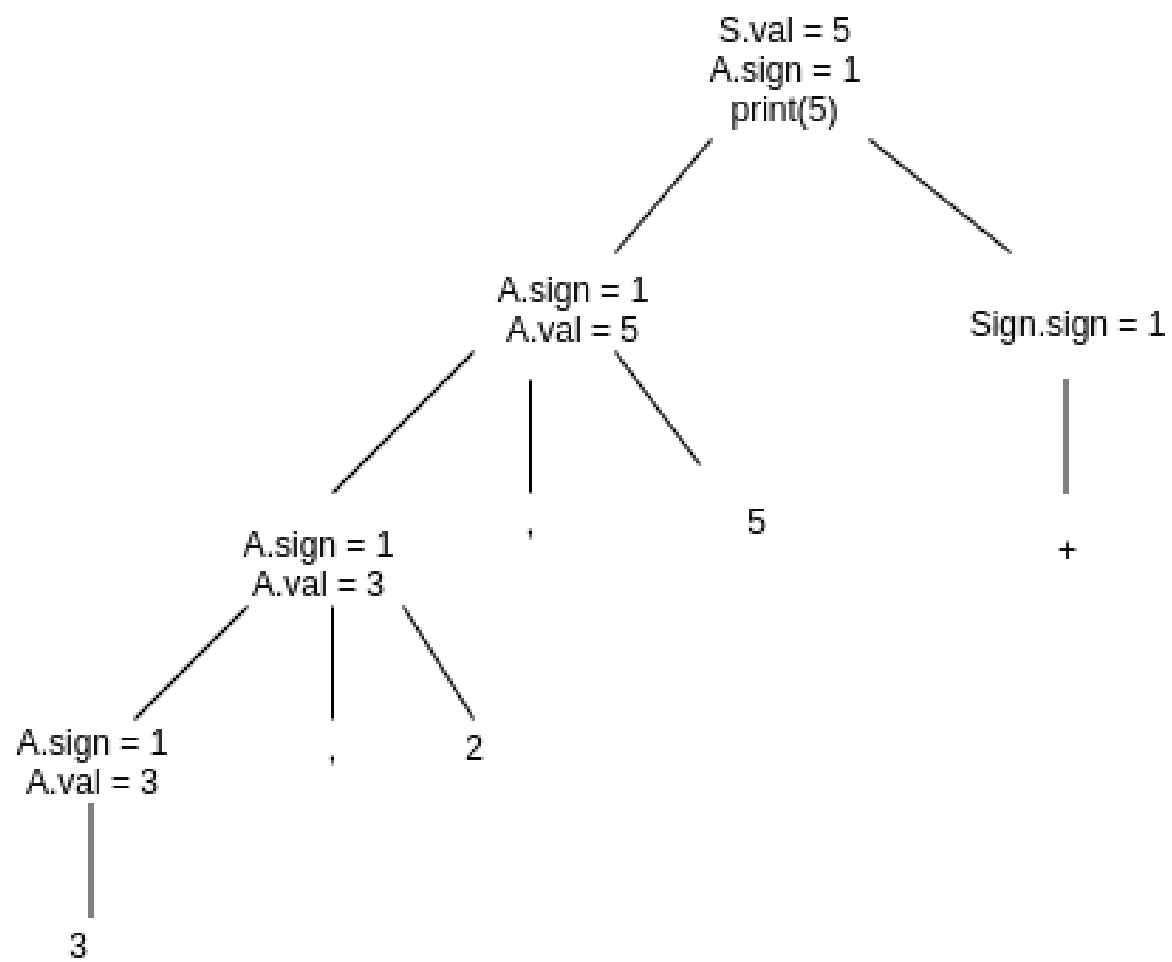
10 11 12

مسئله ۲. SDT

پاسخ.

- اگر آخر رشته - باشد می آید کمترین عدد مجموعه را حساب می کند و در صورتی که + باشد بیشترین عدد را در مجموعه محاسبه می کند.

- $\text{synthesized} \Rightarrow \{S.val, A.val, \text{Sign.sign}\}$
 $\text{inherited} \Rightarrow \{A.sign\}$



•

مسئله‌ی 3. Syntax Graph

پاسخ.
اصلاح گراف:

- روال مفهومی @push را روی یال s8 به s9 قرار می‌دهیم. (id/@push)
- روال مفهومی @add را روی یال s3 به s2 قرار می‌دهیم. (T/@add)
- روال مفهومی @mult را روی یال s7 به s6 قرار می‌دهیم. (F/@mult)

گرامر:

- $E \rightarrow TE'$
- $E' \rightarrow +TE'$
- $E' \rightarrow \lambda$
- $T \rightarrow FT'$
- $T' \rightarrow *FT'$
- $T \rightarrow \lambda$
- $F \rightarrow id$
- $F \rightarrow (E)$

token	remain	PS	SS
	id + id	\$	
id	+ id	E \$	
id	+ id	T E' \$	
id	+ id	F T' E' \$	
id	+ id	@push id T' E' \$	
id	+ id	id T' E' \$	
+	id	T' E' \$	id
+	id	E' \$	id
+	id	+ T @add E' \$	id
id		T @add E' \$	id
id		F T' @add E' \$	id
id		@push id T' @add E' \$	id
id		id T' @add E' \$	id
		T' @add E' \$	id id
		@add E' \$	id id
		E' \$	t
		\$	t

مسئله‌ی 4. single pass vs multi pass

پاسخ. در کامپایلر single pass از آنجا که نمی‌توانیم نسخه پشتیبان تهیه و پردازش کنیم، دوباره دستور زبان باید محدود یا ساده شود. همچنین یکی دیگر از معایب آن این است که قابل حمل نیست. در مقابل کامپایلر multi pass قابل حمل است و حافظه کمتری می‌گیرد ولی در عوض کندتر است

ب: type checking در زمان اجرا دقیق‌تر است اما از آنجایی که کاری که باید در زمان کامپایل انجام می‌دادیم را داریم در زمان اجرا انجام می‌دهیم کارایی کمتری دارد.

مسئله‌ی 5. Type Checking

پاسخ.

```
List list => list : List<Obj>
new ArrayList => list : ArrayList<Obj>
List contains Array list => valid
```

```
"ABCD" is string (i)
list => List of Object
add: Obj => void ==> valid
26 : Integer
```

Object contains Integer

add : valid

(string) list.get(0) : string

String string1 =(string) x : valid

get: Integer -> string

string string2 = (string) x : valid

ب: دیگر نیازی نبود بررسی کنیم که obj شامل string هست یا خیر. در نتیجه لازم نبود cast شدن به بررسی object شود. در خط ۱۰ چون string شامل int نمی شود به ارور ArgumentMissmatch برمی خوریم. و در خط ۱۴ به ارور برمی خوریم ClassCastException