تمرین‌های برنامه‌نویسی درس اصول طراحی کامپایلر

کامپایلر تسلنگ: گام اول

در گام اول از تمرین عملی درس طراحی کامپایلر، باید یک تحلیلگر لغوی بنویسید. در این گام باید برنامه‌ای بنویسید که با خواندن یک فایل در زبان تسلنگ از ورودی استاندارد، واژه‌های (Tokens) آن را چاپ کند. برای مثال، کد تسلنگ زیر را در نظر بگیرید:

```
function example(Array A) returns Int:
      return (arrayLength(A) + 1);
end
```

برای مثال، اگر این قطعه کد به عنوان ورودی داده شود، برنامه‌ی شما باید هر واژه را در یک خط خروجی چاپ نماید:

```
function
example
(
Array
A
)
returns
Int
:
return
(
arrayLength
(
A
)
+
1
)
;
end
```

دقت کنید که واژه‌ها ممکن است با فاصله جدا نشده باشند. این تمرین تا تاریخ ۱۴۰۰/۸/۱۴ قابل انجام است.

کامپایلر تسلنگ: گام دوم

در گام دوم از تمرین عملی درس طراحی کامپایلر، تحلیل نحوی را انجام می‌دهید. در این گام برنامه‌ای می‌نویسید که با خواندن یک فایل تسلنگ از ورودی استاندارد و تحلیل نحوی آن، پیغام‌هایی را چاپ می‌کند. دقت کنید که تجزیه‌ی برنامه‌ی ورودی الزامی است و منطق برنامه باید به کمک عملیات مفهومی نوشته شود. در این پیغام‌ها چند نوع خطا چاپ کنید: یک) فراخوانی یک تابع با تعداد نادرست پارامترها، دو) فراخوانی یک تابع با نوع نادرست پارامترها، سه) برگرداندن مقداری با نوع اشتباه از یک تابع و چهار) دسترسی به متغیرهایی که تعریف نشده‌اند.

```
1     function find(Array A, Int x) returns Int:
2           val Int n;
3           i = 0;
4           foreach (n of A):
5                 if (n == k):
6                       return i;
7                 end
8                 i = i + 1;
9           end
10          return -1;
11    end
12
13    function main() returns Int:
14          val Array A;
15          val Int a;
16          A = createArray(3);
17          A[0] = 3;
18          A[1] = 8;
19          A[2] = 5;
20          printInt(find(A, a));
21          printInt(find(A));
22          printInt(find(a, A));
23          return A;
24    end
```

برنامه‌ی شما پس از خواندن این فایل باید خط‌های زیر را چاپ نماید.

```
3: identifier "i" is not defined!
5: identifier "k" is not defined!
21: function "find" expects 2 arguments but only 1 given!
22: wrong type for argument 1 of "find"!
22: wrong type for argument 2 of "find"!
23: returning a value with wrong type from "main"!
```

این تمرین تا تاریخ ۱۴۰۰/۹/۲۶ قابل انجام است.

کامپایلر تسلنگ: گام سوم

در گام سوم از تمرین عملی درس طراحی کامپایلر، برنامه‌ای می‌نویسید که با خواندن یک فایل تسلنگ از ورودی استاندارد، کد میانی آن را تولید می‌کند. برای نمونه، کد زیر را که در زبان تسلنگ است در نظر بگیرید.

```
function sum3(Int a, Int b, Int c) returns Int:
      val Int sum;
      sum = a + b + c;
      return sum;
end

function main() returns Int:
      val Int a;
      val Int b;
      val Int c;
      a = getInt();
      b = getInt();
      c = getInt();
      printInt(sum3(a, b, c));
      return 0;
end
```

برنامه‌ی شما پس از خواندن این فایل باید کد میانی تسلنگ را تولید کند. یک خروجی نمونه برای این دو تابع در ادامه نشان داده می‌شود.

```
proc  sum3
   add   r0, r0, r1
   add   r0, r0, r2
   ret

proc  main
   call  iget, r3
   call  iget, r1
   call  iget, r2
   call  sum3, r3, r1, r2
   call  iput, r3
   mov   r0, 0
   ret
```

این تمرین تا تاریخ ۹۹/۱۰/۳۰ قابل انجام است. به نکته‌های زیر توجه کنید:

- برای بررسی درستی کد تولید شده، با استفاده از برنامه‌ی tsvm کد میانی را اجرا کنید.
- برای خواندن ورودی و خروجی می‌توانید از توابع داخلی TSIR استفاده کنید.
- فرض کنید بردارهای تسلنگ با اندازه‌ی n، یک اشاره‌گر به قسمتی از حافظه با اندازه‌ی $n + 1$ عدد هستند. عدد اول طول بردار و سایر عددها محتویات بردار را نشان می‌دهند.