

# فصل سوم

## پیشته

نجمه منصوری

## ساختمانهای پویا (Dynamic)

برخی از مزایای استفاده از آرایه ها :

- ۱- فضای بهم پیوسته نیاز دارند.
- ۲- حذف یک عضو ، شیفت دادن عناصر بعدی را به سمت پائین می طلبد.
- ۳- درج یک عضو نیز شیفت دادن عناصر بعدی را به سمت بالا می طلبد.
- ۴- آرایه ها ایستا هستند.

# پشته

- پشته. لیستی که در آن عمل درج و حذف از یک طرف به نام بالای پشته انجام می‌شود.
- درج و حذف بر مبنای اصل «آخرین ورودی اولین خروجی»



## اشاره‌گر top و شرایط مرزی (خالی یا پر بودن stack)

در صورتی که از آرایه  $[1 .. n]$  برای نگهداری عناصر پشته استفاده شود، شرایط مرزی به صورت زیر خواهد بود:

پشته پر (Full)	پشته خالی (empty)	وضعیت top
$\text{top} = n$ $\vdots$	 $\text{top} = 0$	آخرین خانه پر (پیش فرض)
$\text{top} = n + 1$ $\vdots$	 $\text{top} = 1$	اولین خانه خالی

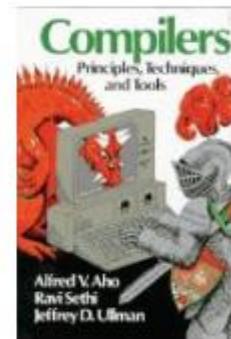
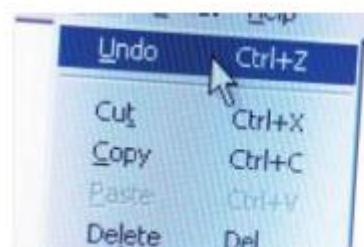
نکته مهم: همیشه به طور پیش فرض  $\text{top}$  به آخرین خانه پر اشاره می‌کند.

## الگوریتم‌های pop (حذف) و push (درج)

الگوریتم(حذف)	الگوریتم(درج)	وضعیت top
<pre> procedure pop ( var item : data type); begin   if top = 0 then     write ('stack is Empty')   else     begin       item := stack [top];       top := top - 1;     end;   </pre>	<pre> procedure push ( var item : data type); begin   if top = n then     write ('stack is Full')   else     begin       top := top + 1;       stack [top] := item;     end;   </pre>	<p style="text-align: center;">آخرین پر</p> <p style="text-align: center;">3</p>
<pre> procedure pop ( var item : data type); begin   if top = 1 then     write ('stack is Empty')   else     begin       top := top - 1;       item := stack [top];     end;   </pre>	<pre> procedure push ( var item : data type); begin   if top = n+1 then     write ('stack is Full')   else     begin       stack [top] := item;       top := top + 1;     end;   </pre>	<p style="text-align: center;">اولین خالی</p> <p style="text-align: center;">4</p>

# کاربردهای پیشنهادی

- تجزیه در کامپایلر
  - عملگر `undo` در یک واژه‌پرداز
  - دکمه‌ی برگشت در یک مرورگر وب
  - پیاده‌سازی فراخوانی توابع در یک کامپایلر
- ارزیابی عبارت‌های محاسباتی (prefix – postfix – infix)

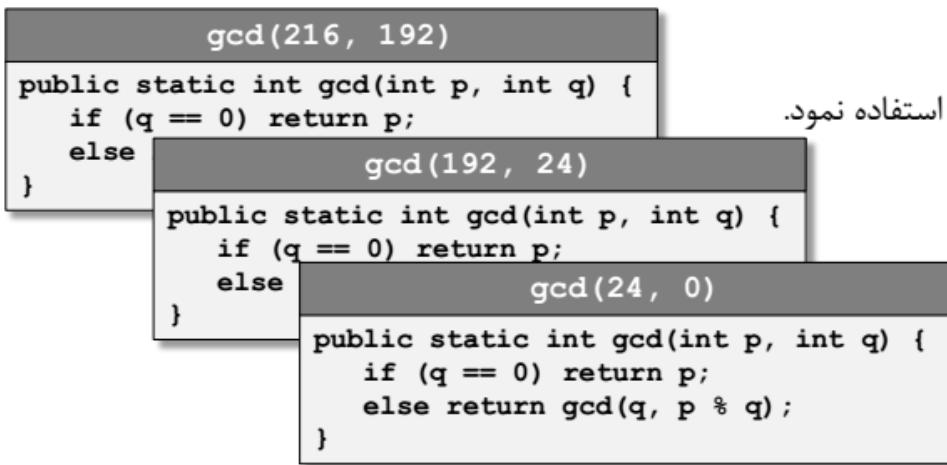


# فراخوانی توابع

□ چگونه کامپایلر اجرای توابع را ممکن می‌سازد؟

□ فراخوانی تابع: ذخیره مقادیر متغیرهای محلی و آدرس بازگشت در پشته

□ برگشت: برداشتن آدرس بازگشت و مقادیر متغیرهای محلی از پشته



□ تابع بازگشتی. تابعی که خودش را فراخوانی می‌کند.

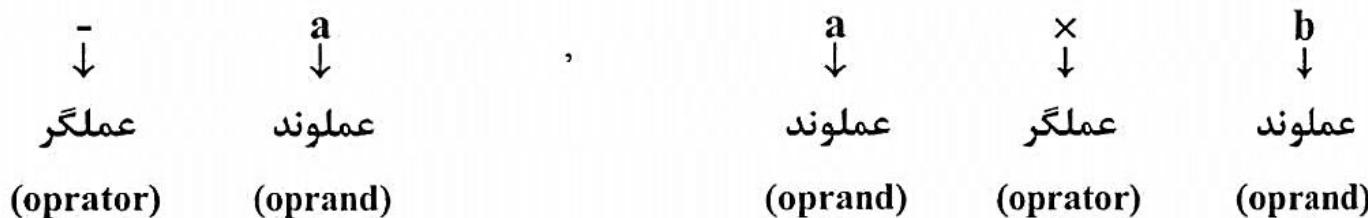
□ همواره می‌توان از پشته برای حذف فراخوانی‌های بازگشتی استفاده نمود.

## ارزشیابی عبارت‌های محاسباتی

هر عبارت محاسباتی با توجه به اولویت عملگرهاش قابل ارزیابی و محاسبه است.

یادآوری ۱:

در هر عبارت محاسباتی:



یادآوری ۲:

با توجه به یادآوری ۱ دیده می‌شود که در عبارت‌های محاسباتی دو نوع عملگر وجود دارد:

۱- یکانی (unary)      ۲- باینری (binary)

عملگرهای یکانی مانند: - (منفی) و + (مثبت) فقط به یک عملوند نیاز دارند.

عملگرهای دودویی مانند:  $\times$  (ضرب)،  $/$  ( تقسیم)،  $+$  (جمع) و  $-$  (تفريق) و  $\uparrow$  (توان) به دو عملوند نیاز دارند.

## اولویت عملگرها

به طور کلی صرف نظر از هر زبان برنامه‌سازی اولویت عمومی عملگرها را می‌توان به شرح زیر در نظر گرفت:

اولویت	نام عملگر	توضیحات
۱	( )	
۲	- (منفی)، + (مثبت)	
۳	↑ یا ^ (توان)	اولویت توان‌های متوالی از راست به چپ
۴	* (ضرب)، / (تقسیم)	هم اولویت، اولویت از چپ به راست
۵	+ (جمع)، - (تفريق)	هم اولویت، اولویت از چپ به راست.

دقت کنید:

- ۱- در هر عبارت محاسباتی اگر تعداد عملوندها ۱ واحد بیشتر از تعداد عملگرها باشد، در آن عبارت همه عملگرها دودویی هستند.
- ۲- در هر عبارت محاسباتی اگر تعداد عملگرها بیشتر یا مساوی تعداد عملوندها باشد. در آن عبارت حتماً عملگر یکانی وجود دارد.

مثال : اولویت‌بندی عبارت‌های زیر را در نظر بگیرید:  
عبارت ۱ :

$$\underbrace{a * b}_{3} - \underbrace{c \wedge d \wedge e}_{\substack{1 \\ 2}} + f$$

$$\underbrace{\quad\quad\quad}_{4}$$

$$\underbrace{\quad\quad\quad}_{5}$$

عبارت ۲ :

$$a / \underbrace{(b + c)}_{1} - e * \underbrace{(f + g)}_{\substack{2 \\ 4}}$$

$$\underbrace{\quad\quad\quad}_{3}$$

$$\underbrace{\quad\quad\quad}_{5}$$

## درخت عبارت محاسباتی (درخت پارس)

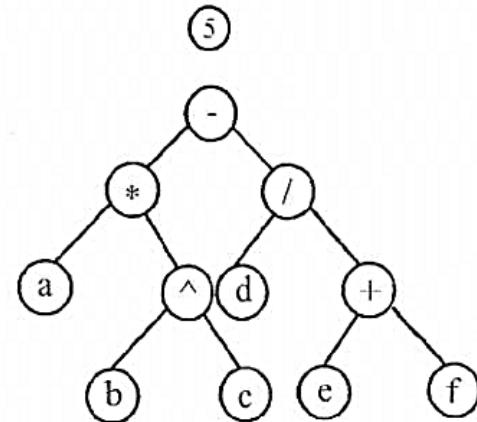
برای تشکیل درخت هر عبارت محاسباتی به صورت زیر عمل می کنیم:

- ۱- ابتدا عبارت را بر حسب اولویت عملگرهایش اولویت‌بندی (شماره گذاری) می کنیم.
- ۲- ریشه درخت، عملگر با کمترین اولویت (بیشترین شماره) است.
- ۳- ریشه زیر درختان چپ و راست نیز به همین ترتیب عملگر با کمترین اولویت (بیشترین شماره) در چپ و راست ریشه درخت خواهد بود.
- ۴- برگ‌های درخت عملوند و سایر گره‌ها (افرزندی یا فرزندی)، عملگر خواهند بود.

عبارت محاسباتی	اولویت بندی عبارت	درخت عبارت
$a * b - c / d$	$\frac{a * b - c}{d}$ <span style="display: flex; justify-content: space-around;"> <span>1</span> <span>2</span> <span>3</span> </span>	<pre> graph TD     Root(-) --- Mult(*)     Root --- Div(/)     Mult --- a[a]     Mult --- b[b]     Div --- c[c]     Div --- d[d]   </pre> <p>کم اولویت ترین عملگر - (تفريق) بوده که در ریشه قرار گرفته است.</p>

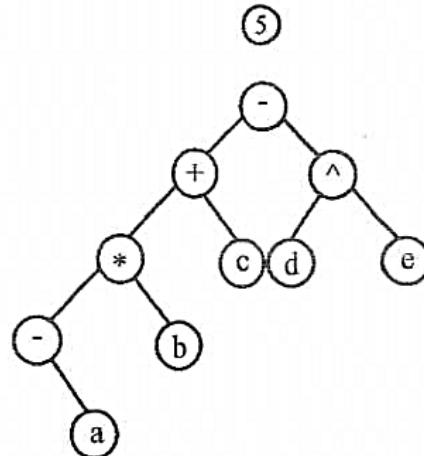
$$a * b^c - d / (e + f)$$

$$\underbrace{a * \underbrace{b^c}_{2}}_{3} - \underbrace{d / \underbrace{(e + f)}_{4}}_{5}$$



$$- a * b + c - d ^ e$$

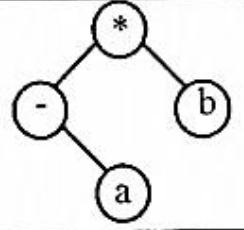
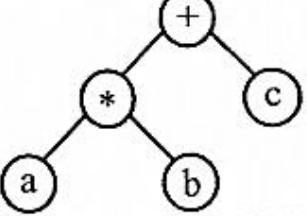
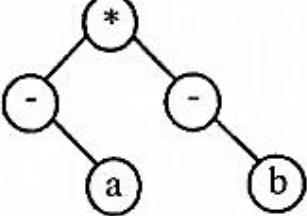
$$\begin{array}{c} - a * b + c - d ^ e \\ \overbrace{\quad\quad\quad}^1 \quad \overbrace{\quad\quad\quad}^2 \\ \quad \quad \quad \overbrace{\quad\quad\quad}^3 \quad \quad \quad \overbrace{\quad\quad\quad}^4 \\ \quad \quad \quad \quad \quad \overbrace{\quad\quad\quad\quad\quad}^5 \end{array}$$



دیده می شود که عملوند مربوط به عملگر یکانی (-)، یعنی a سمت راست آن قرار گرفته است.

## رابطه بین تعداد گره‌ها و عملگرهاي يکاني (Unary) (Unary)

به عبارات و درختان زير دقت کنيد:

عبارت	درخت عبارت	تعداد گره	توضیحات
$- a * b$		4 گره	1 عملگر يکاني داريم
$a * b + c$		5 گره	0 عملگر يکاني داريم (عملگر يکاني نداريم)
$- a * - b$		5 گره	2 عملگر يکاني داريم

نتیجه :

توضیحات	تعداد عملگر یکانی (Unary)	تعداد گره‌های درخت عبارت
حداقل ۱ عملگر یکانی داریم	۱, ۳, ۵, ... (فرد)	زوج
ممکن است عملگر یکانی نداشته باشیم	۰, ۲, ۴, ... (زوج)	فرد

مثال : تعداد گره‌های یک درخت دودویی که یک عبارت ریاضی را نمایش می‌دهد ۱۴ می‌باشد. عملگرهای این عبارت، دودویی یا یکانی (unary) می‌باشند. کدامیک از گزینه‌های زیر درست است؟

- ۱) این عبارت حتماً تعداد فردی عملگر دودویی دارد.
- ۲) این عبارت حتماً تعداد زوجی عملگر یکتاوی دارد.
- ۳) این عبارت نمی‌تواند عملگر دودویی داشته باشد.
- ۴) حداقل یک عملگر یکانی در این عبارت وجود دارد.

حل : گزینه ۴ درست است.

## عبارت infix (میانوندی)

در هر عبارت عمومی محاسباتی هر عملگر دودویی بین عملوندهایش قرار می‌گیرد، که به این نحوه قرار گرفتن عملگرها در بین عملوندها روش میانوندی گفته می‌شود.

$a \times b \rightarrow$  بین عملوندهای  $a$ ,  $b$  قرار دارد.

در هر عبارت میانوندی (infix) اولویت عملگرها مطرح بوده و از ( ) برای تغییر اولویت عملگرها استفاده می‌شود.

در هر عبارت میانوندی (infix) اولویت عملگرها با توجه به جدول اولویت عملگرها مشخص می‌شود.

## عبارت postfix (پسوندی)

در این روش هر عملگر بعد از عملوندهایش قرار می‌گیرد.

عملگر  $\times$  بعد از عملوندهایش قرار گرفته است.  $\rightarrow ab \times$

در هر عبارت postfix، اولویت عملگرها مطرح نبوده و ( ) وجود ندارد.

به روش postfix روش لهستانی معکوس یا polish وارونه یا RPN (Reverse polish Notation) نیز می‌گویند.

## عبارت prefix (پیشوندی)

در این روش هر عملگر قبل از عملوندهایش قرار می‌گیرد.

عملگر  $\times$  قبل از عملوندهایش قرار گرفته است.  $\rightarrow ab$

در هر عبارت prefix، اولویت عملگرها مطرح نبوده و ( ) وجود ندارد.

به روش prefix روش لهستانی یا روش polish نیز می‌گویند چون اولین بار این روش توسط ریاضیدان لهستانی معرفی شد. به این دلیل است که به روش postfix روش لهستانی وارونه گفته می‌شود.

(الف)

$$\text{infix: } a \times b + c \rightarrow ((a \times b)^* + c)^+ \longrightarrow \text{postfix: } a, b, \times, c, +$$

(ب)

$$\text{infix: } a \times (b + c) - g / d \longrightarrow \left( \left( a \times (b + c)^+ \right)^* - (g / d)' \right)^- \rightarrow \text{postfix: } a, b, c, +, \times, g, d, /, -,$$

(ج)

$$\text{infix: } \sqrt{b^2 - 4ac} = (b \uparrow 2 - 4 \times a \times c) \uparrow (1/2) \rightarrow \left( \left( (b \uparrow 2)^{\dagger} - ((4 \times a)^* \times c)^* \right)^{\dagger} \uparrow (1/2)' \right)^{\dagger}$$

$$\rightarrow \text{postfix: } b, 2, \uparrow, 4, a, \times, c, \times, -, 1, 2, /, \uparrow$$

(د)

$$\text{infix: } a \times b + c \uparrow d \uparrow e - f \rightarrow \left( \left( (a \times b)^* + \left( c \uparrow (d \uparrow e)^{\dagger} \right)^{\dagger} \right)^+ - f \right)^-$$

$$\rightarrow \text{postfix: } a, b, \times, c, d, e, \uparrow, \uparrow, +, f, -$$

(هـ)

$$\text{infix: } a^* - b \uparrow c \uparrow d - e / f \rightarrow \left( \left( a^* \left( (-b)^- \uparrow (c \uparrow d)^{\dagger} \right)^{\dagger} \right)^* - (e / f)' \right)^-$$

$$\rightarrow \text{postfix: } a, b, -, c, d, \uparrow, \uparrow, *, e, f, /, -$$

(الف)

$$\text{infix: } a * b + c \rightarrow {}^+({}^*(a * b) + c) \rightarrow \text{prefix: } +, *, a, b, c$$

(ب)

$$\text{infix: } a * (b + c) - g / d \rightarrow {}^-({}^*(a * {}^+(b + c)) - {}'(g / d)) \rightarrow \text{prefix: } -, *, a, +, b, c, /, g, d$$

(ج)

$$\begin{aligned}\text{infix: } \sqrt{b^2 - 4ac} &= (b \uparrow 2 - 4 * a * c) \uparrow (1 / 2) \rightarrow {}^+({}^-({}^\uparrow(b \uparrow 2)) - {}^*({}^*(4 * a) * c)) \uparrow {}'(1 / 2) \\ \rightarrow \text{prefix: } &\uparrow, -, \uparrow, b, 2, *, *, 4, a, c, /, 1, 2\end{aligned}$$

(د)

$$\begin{aligned}\text{infix: } a * b + c \wedge d \wedge e - f &\rightarrow {}^-({}^+({}^*(a * b) + {}^\uparrow(c \uparrow {}^\uparrow(d \uparrow e))) - f) \\ \rightarrow \text{prefix: } &- , + , * , a , b , \uparrow , c , \uparrow , d , e , f\end{aligned}$$

(هـ)

$$\begin{aligned}\text{infix: } a * -b \uparrow c \uparrow d - e / f &\rightarrow {}^-({}^*({}^a * {}^\uparrow({}^-(-b) \uparrow {}^\uparrow(c \uparrow d))) - {}'(e / f)) \\ \rightarrow \text{prefix: } &- , * , a , \uparrow , - , b , \uparrow , c , d , / , e , f\end{aligned}$$



برای تبدیل یک عبارت پسوندی (postfix) به میان‌وندی (infix) با استفاده از روش پرانتزگذاری، ابتدا عبارت را از چپ به راست بیمایش کرده به هر عملگر که رسیدیم از سمت چپ آن، دو عملوند را در پرانتز قرار می‌دهیم تا کل عبارت پرانتزگذاری شود، سپس هر عملگر را بین عملوند‌هایش انتقال می‌دهیم.

معادل میان‌وندی عبارت پسوندی  $-abc + *de$  چیست؟

$$a * (b + c) - d / e \quad (1)$$

$$a + b * c / (d - e) \quad (2)$$

$$a - (b / c) * (d + e) \quad (3)$$

$$a - (b * d) / (c + e) \quad (4)$$

پاسخ: گزینه «۴» با استفاده از روش پرانتزگذاری، داریم:

$$((a(bc+)*)(de/-)) \Rightarrow ((a(b+c)*)(de/-)) \Rightarrow ((a*(b+c))(de/-)) \Rightarrow ((a*(b+c))(d/e)-) \Rightarrow a * (b + c) - d / e$$

برای تبدیل یک عبارت پیشوندی (prefix) به میان‌وندی (infix) با استفاده از روش پرانتزگذاری، ابتدا عبارت را از راست به چپ پیمایش کرده به هر عملگر که رسیدیم از سمت راست آن، دو عملوند را در پرانتز قرار می‌دهیم تا کل عبارت پرانتزگذاری شود، سپس هر عملگر را بین عاملوند هایش انتقال می‌دهیم.

$+ - AB / B + CD$

معادل infix عبارت prefix روبرو کدام است؟

$$A + B - B / (C + D) \quad (4)$$

$$A - (B + B) / (C + D) \quad (3)$$

$$A - B + B / (C + D) \quad (2)$$

$$A - B + B / C + D \quad (1)$$

پاسخ: گزینه «۲» با استفاده از روش پرانتزگذاری داریم:

$$(+(-AB)(/ B(+CD))) = (+(A - B)(/ B(C + D))) = (+(A - B)(B / (C + D))) = ((A - B) + (B / (C + D))) = A - B + B / (C + D)$$

اگر  $d = 10$  ،  $c = 8$  ،  $b = 4$  ،  $a = 2$  باشد، ارزش عبارت پسوندی  $ab^*c + da - /$  چیست؟

2 (۴)

1 (۳)

-1 (۳)

-2 (۱)

پاسخ: گزینه «۴» ابتدا عبارت پسوندی را به میان‌وندی تبدیل می‌کنیم، داریم:

$$(((ab^*)c+)(da-)/) \Rightarrow (((a * b)c+)(da-)/) \Rightarrow ((a * b + c)(da-)/) \Rightarrow (a * b + c)/(d - a)$$

$$(2 * 4 + 8) / (10 - 2) = 16 / 8 = 2$$

سپس با جایگزینی اعداد، نتیجه حاصل می‌شود:

معادل میان‌وندی (infix) عبارت پسوندی (postfix) روبرو چیست؟

$A ^ (B - C) / (D + E)$  (۴)

$A ^ B - C / (D + E)$  (۳)

$A ^ (B - C) / D + E$  (۲)

$A ^ B - C / D + E$  (۱)

پاسخ: گزینه «۲» با استفاده از روش پرانتر-گذاری داریم:

$$(((A(BC-)^)D/E)+) = (((A(B-C)^)D/E)+) = (((A ^ (B-C))D/E)+)$$

$$= (((A ^ (B-C))/D)E+) = (((A ^ (B-C))/D) + E) = A ^ (B-C) / D + E$$

معادل پسوندی (postfix) عبارت میانو ندی  $(A/B^C)*D+E$  کدام است؟

$$AB/C^DE*+(٤)$$

$$ABC^/D*E+(٣)$$

$$ABC/^DE*+(٢)$$

$$AB^C/DE*(١)$$

پاسخ: گزینه «۳» با استفاده از روش پرانتزگذاری، داریم:

$$(((A/(B^C))*D)+E) \Rightarrow (((A/(BC^))*D)+E) \Rightarrow (((ABC^/)*D)+E) \Rightarrow ((ABC^/D*)+E) \Rightarrow ABC^/D*E+$$

$$a+b*(c/(d+e))*f$$

می خواهیم عبارت میانو ندی (infix) زیر را به کمک پشته به عبارت پسوندی (postfix) تبدیل کنیم:

برای این کار چند بار باید پردازه push را برای گذاشت اجزاء این عبارت در پشته، فراخوانی کنیم؟

7 (٤)

9 (٣)

5 (٢)

15 (١)

پاسخ: گزینه «۴» با توجه به الگوریتم ارائه شده در این بخش، تمام عملگرها و پرانتز بازها باید یک بار در پشته قرار گیرند. با شمارش پرانتز بازها و عملگرها، تعداد فراخوانی پردازه Push مشخص می شود.

$$x+y/(u-t*w)*y$$

معادل پسوندی Postfix عبارت ریاضی میانو ندی Infix روبرو کدام است؟

$$tw*u-y/x+y*(٤)$$

$$utw*-xy/+y*(٣)$$

$$xyutw*-/y*+(٢)$$

$$xyutwy*-*/+(١)$$

پاسخ: گزینه «۲» ابتدا پرانتزگذاری کرده و سپس عملگرها را به پرانتز بسته ها منتقل می کنیم:

$$(x+((y/(u-(t*w)))*y)) \Rightarrow (x+((y/(u-(tw*)))*y)) \Rightarrow (x+((y/(utw*-*))*y))$$

$$\Rightarrow (x+((yutw*-/)*y)) \Rightarrow (x+(yutw*-/y*)) \Rightarrow xyutw*-/y*+$$

### نتیجه:

- ۱- در تبدیل عبارات infix ، postfix و prefix به یکدیگر ترتیب عملوندها به هیچ وجه عوض نمی‌شود.
- ۲- در هر عبارت postfix همیشه سمت راست عبارت عملگر است.
- ۳- در هر عبارت prefix همیشه سمت چپ عبارت عملگر است

$a \times b + c \rightarrow \text{infix}$

$a b \times c + \rightarrow \text{postfix}$

$+ \times a b c \rightarrow \text{prefix}$

## تبدیل postfix به infix

برای این تبدیل به ترتیب زیر عمل می‌کنیم:

الف) عبارت را از سمت چپ پیمایش می‌کنیم.

ب) عملوندها را در خروجی می‌نویسیم.

ج) به هر پرانتز ( ) که رسیدیم آن را به راحتی داخل Stack قرار می‌دهیم.

د) به هر عملگر که رسیدیم به شرطی که اولویت آن عملگر از عملگر بالای stack بیشتر باشد، آن را داخل stack قرار می‌دهیم، در غیر این صورت آنقدر از بالای stack عملگر خارج کرده و در خروجی می‌نویسیم تا یا stack خالی شده و یا به عملگری بررسیم که بتوانیم عملگر مورد نظر را روی آن در stack قرار دهیم.

## یادآوری:

باتوجه به مورد (د)، عملگر + نمی‌تواند روی + یا - قرار گیرد، همین‌طور - روی + یا - و / روی \* یا / و \* روی \* یا / نمی‌توانند قرار گیرند. اما توان  $(\uparrow)$  روی توان  $(\uparrow)$  می‌تواند قرار گیرد، چون اولویت توان‌های پشت سرهم از راست به چپ بررسی می‌شود.

ه) اگر بالای stack پرانتز ')' باشد هر عملگری به راحتی روی آن قرار می‌گیرد.

و) به هر پرانتز ')' که رسیدیم آنقدر از stack عملگر خارج کرده و در خروجی می‌نویسیم تا به ')' برسیم در این وضعیت '()

ر) زمانی که به انتهای عبارت رسیدیم در صورتی که، stack خالی نباشد، آن را به طور کامل در خروجی می‌نویسیم.

**مثال ۱:** حداقل اندازه پشته برای تبدیل عبارت میانوندی (postfix) کدام است؟

3 (f)

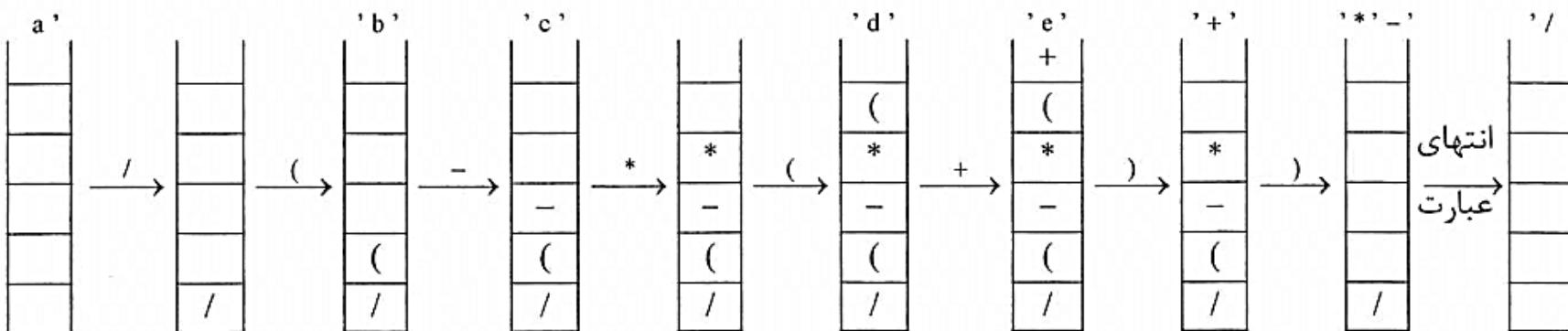
6 (r)

5 (Y)

4 (V)

حل: گزینه ۳ درست است.

از چپ به راست عبارت را پیمایش می‌کنیم.

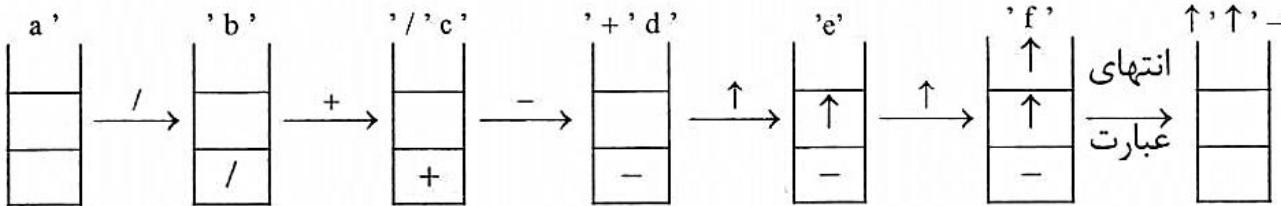


نتیجه را از چپ به راست می‌نویسیم: / - , \* , + , e , d , c , b , a

حداقل فضای مورد نیاز :

**تعداد = push : تعداد عملگرها + تعداد(')**

مثال ۲: حداقل اندازه پشته برای تبدیل عبارت میانوندی  $a / b + c - d \uparrow e \uparrow f$  به پسوندی کدام است؟



نتیجه از چپ به راست:  $a, b, /, c, +, d, e, f, \uparrow, \uparrow, -, \uparrow, \uparrow$  خواهد بود.

حداقل فضای مورد نیاز: ۳

تعداد **push** = تعداد **pop** : تعداد عملگرها + تعداد '()'

## تبدیل prefix به infix

برای این تبدیل به ترتیب زیر عمل می‌کنیم:

✓ الف) عبارت را از سمت راست پیمایش می‌کنیم.

ب) عملوندها را در خروجی می‌نویسیم.

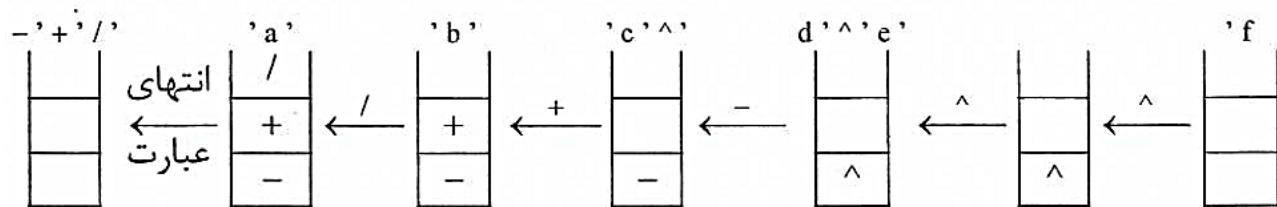
ج) به هر پرانتز '()' که رسیدیم آن را به راحتی داخل stack قرار می‌دهیم.

✓ د) به هر عملگر که رسیدیم به شرطی که اولویت آن عملگر از عملگر بالای stack بیشتر یا مساوی باشد، آن را داخل stack قرار می‌دهیم، در غیر این صورت آن قدر از بالای stack عملگر خارج کرده و در خروجی می‌نویسیم تا یا stack خالی شده و یا به عملگری بررسیم که بتوانیم عملگر مورد نظر را روی آن در stack قرار دهیم.

## یادآوری:

- باقجه به مورد (د)، عملگر - و + می‌توانند روی + یا - قرار گیرند، همین‌طور \* و / می‌توانند روی \* یا / قرار گیرند. اما توان  $\uparrow$  روی توان  $\uparrow$  نمی‌توانند قرار گیرد. چون اولویت توان‌های پشت سرهم از راست به چپ بررسی می‌شود.
- ۵) اگر بالای stack پرانتز ')' باشد هر عملگری به راحتی روی آن قرار می‌گیرد.
- ۶) به هر پرانتز ')' که رسیدیم آنقدر از stack عملگر خارج کرده و در خروجی می‌نویسیم تا به ')' برسیم در این وضعیت ')' و ')' با هم خنثی می‌شوند.
- ۷) زمانی که به ابتدای عبارت رسیدیم (چون عبارات را از راست به چپ بررسی می‌کنیم) در صورتی که stack خالی نباشد، آن را به طور کامل در خروجی می‌نویسیم.

مثال ۱: حداقل اندازه پشته برای تبدیل عبارت میانوندی  $a / b + c - d ^ e ^ f$  به عبارت پیشوندی معادل کدام است؟



نتیجه از چپ به راست:  $- , + , / , a , b , c , ^ , d , ^ , e , f$  خواهد بود.

حداقل فضای مورد نیاز: ۳

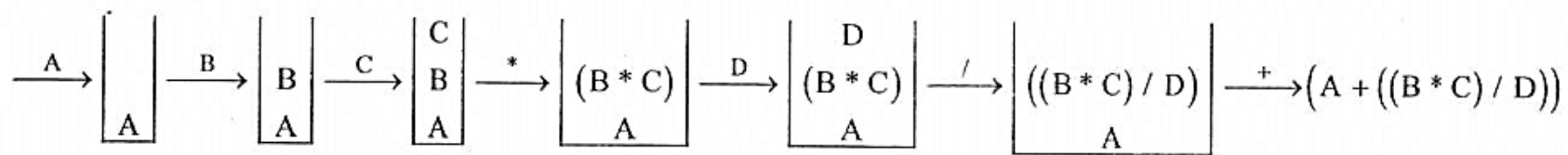
تعداد  $= \text{تعداد push} + \text{تعداد pop}$

## تبدیل عبارات infix و postfix به عبارات prefix

- الف) عبارت را از سمتی بررسی می‌کنیم که با عملوند شروع می‌شود، در postfix از چپ و در prefix از راست عبارت را بررسی می‌کنیم.
- ب) عملوندها را در stack قرار می‌دهیم و با رسیدن به هر عملگر (به جز عملگر آخر) دو عملوند خارج کرده و بعد از محاسبه دوباره حاصل عبارت را داخل stack قرار می‌دهیم. (دقت کنید که دو عملوندی که از stack خارج می‌کنیم از سمت چپ به راست عمل می‌کنیم)
- ج) به عملگر آخر که رسیدیم بعد از خارج کردن دو عملوند آخر و محاسبه آن را دیگر در stack قرار نمی‌دهیم و حاصل را در خروجی می‌نویسیم.

مثال ۱ :

postfix :  $\bar{A}, B, C, *, D, /, +$



حداقل فضای مورد نیاز: 3

تعداد push = pop : تعداد عملگرها  $= 2 \times 3 = 6$

مثال ۲: مینیمم تعداد متغیرهای میانی در محاسبه عبارت جبری  $ab+cd^*/a+$  به صورت postfix است، برابر است با ...

(کارشناسی ارشد - علوم کامپیوتر ۷۹)

4 (۴)

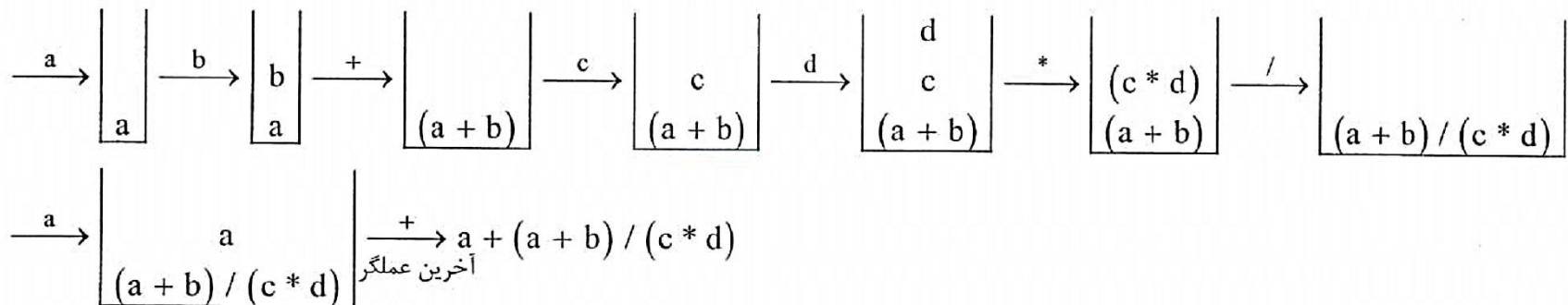
3 (۳)

2 (۲)

1 (۱)

حل: گزینه ۳ درست است،

از سمت چپ بررسی می‌کنیم:



حداقل فضای مورد نیاز: 3

تعداد  $\text{pop} = \text{push}$ : تعداد عملگرهای pop

## خروجی‌های مجاز برای ورودی‌های پشته:

در یک پشته  $n$  تایی در صورتی که داده‌های  $a_1, a_2, \dots, a_n$  به ترتیب وارد Stack شوند، با رعایت قواعد زیر می‌توانیم خروجی‌های مجاز را تعیین کنیم:

- (الف) هر داده به محض ورود می‌تواند از پشته خارج شود.
- (ب) در هر مرحله هرگاه بخواهیم داده‌ای را در خروجی ببینیم که هنوز وارد Stack نشده است می‌بایست داده‌ها را پشت سرهم وارد پشته کنیم تا به داده مورد نظر برسیم سپس داده را وارد پشته کرده و سپس خارج می‌کنیم.
- (ج) در هر مرحله هرگاه بخواهیم داده‌ای را در خروجی ببینیم که پایین Stack است و عنصر بالای پشته نیست /ین عمل غیرمجاز است.

مثال : یک پشته خالی با اعداد از ۱ تا ۶ در ورودی داده شده است. اعمال زیر بر روی پشته قابل انجام هستند:

**PUSH** :: کوچکترین عدد ورودی را برداشته و وارد پشته می کنیم.

**POP** :: عنصر بالای پشته را در خروجی نوشته و سپس آن را حذف می کنیم.

مثال : کدامیک از گزینه های زیر را نمی توان با هیچ ترتیبی از ورودی فوق به دست آورد؟ ( اعداد را از چپ به راست بخوانید.)

1 , 2 , 3 , 4 , 5 , 6

(کارشناسی ارشد دولتی ۷۴)

2,1,5,3,6,4 (۴)

4,3,2,1,6,5 (۳)

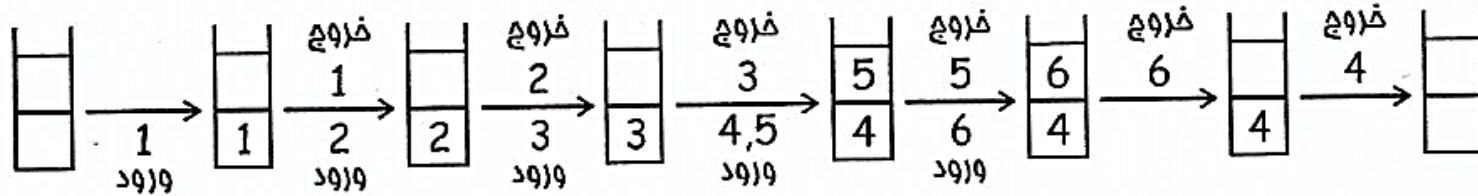
3,2,4,6,5,1 (۲)

1 , 2 , 3 , 5 , 6,4 (۱)

حل: گزینه ۴ درست است.

گزینه ۱ مجاز است زیرا:

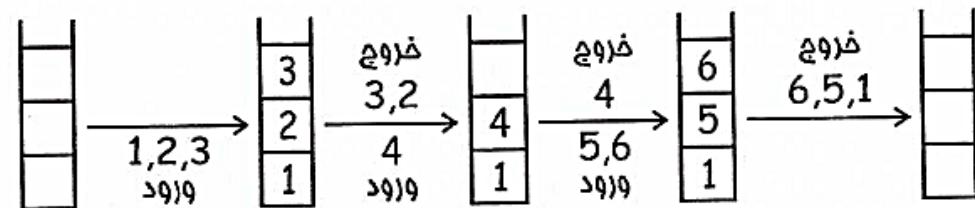
: ورودی ۱, ۲, ۳, ۴, ۵, ۶



: خروجی ۱, ۲, ۳, ۵, ۶, ۴

گزینه ۲ مجاز است. زیرا :

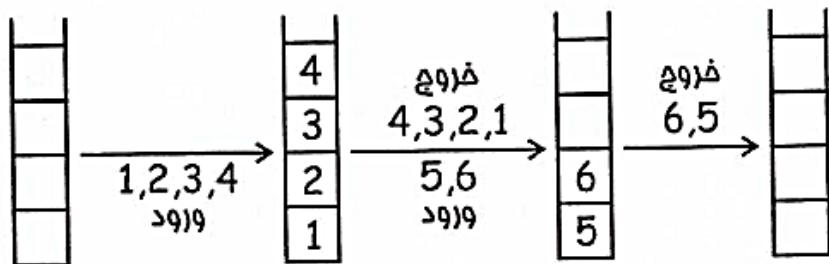
: ورودی ۱, ۲, ۳, ۴, ۵, ۶



: خروج ۳, ۲, ۴, ۶, ۵, ۱

گزینه ۳ مجاز است. زیرا

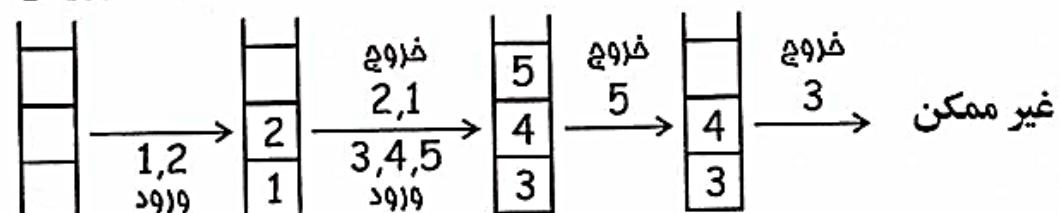
: ورودی ۱, ۲, ۳, ۴, ۵, ۶



: خروج ۴, ۳, ۲, ۱, ۶, ۵

گزینه ۴ امکان‌پذیر نیست زیرا :

: ورودی ۱, ۲, ۳, ۴, ۵, ۶

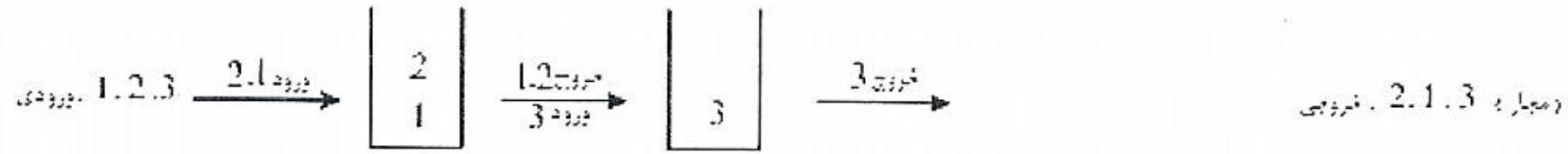
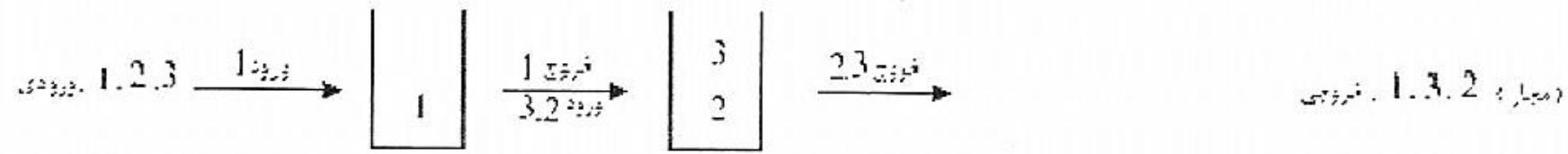
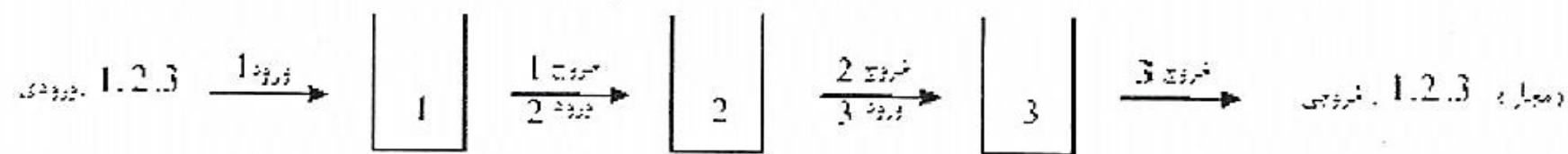


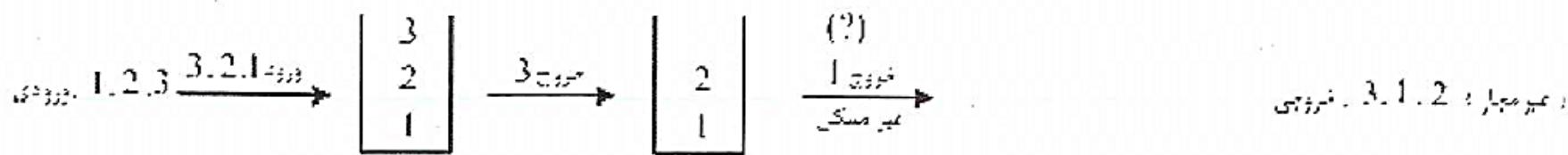
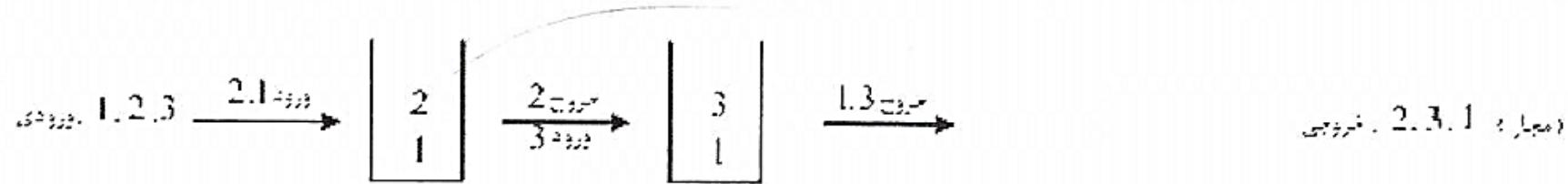
: خروج ۲, ۱, ۵, ۳, ۶, ۴

## تعداد خروجی مجاز

تعداد خروجی های مجاز در یک پشتۀ  $n$  تایی همان عدد کاتالان بوده و بصورت زیر بدست می آید:

$$\frac{\binom{2n}{n}}{n+1}$$





مثال : ورودی  $\overline{A,B,C,D,E,F}$  به یک پشته را در نظر بگیرید، کدام یک از ترتیب های زیر در پشته مجاز (ممکن) است؟

E (مجاز)

C

A

ابتدا A وارد شده سپس B خارج شده C وارد شده سپس D خارج شده E وارد می شود.

F

B

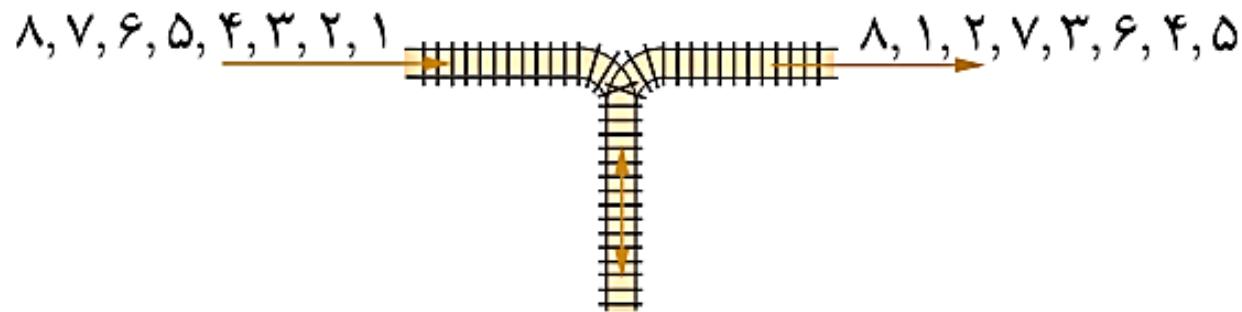
A

(غیرمجاز)

D

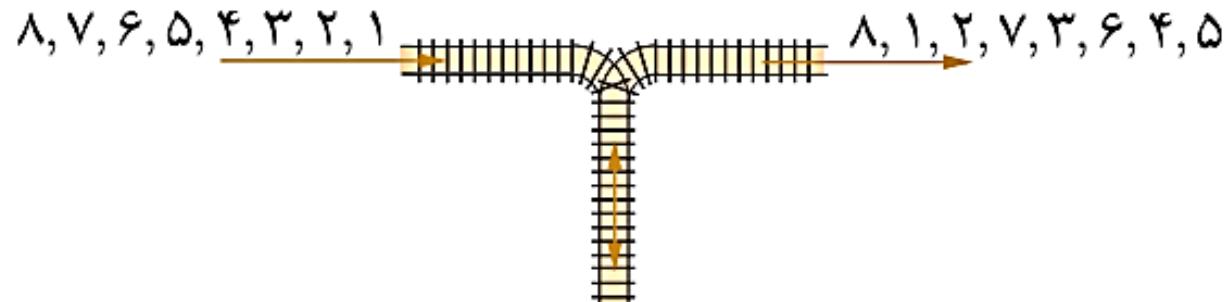
B

A یا باید پایین B قرار گیرد یا قبل از پشته حذف شده باشد.



پشته‌ی قطارها.

$\langle 8, 1, 2, 7, 3, 6, 4, 5 \rangle$  قابل تولید است. چه طور؟



پشته‌ی قطارها.

قابل تولید است. چه طور؟

Push, Push, Push, Push, Push, Pop, Pop, Push, Pop, Pop, Push, Pop, Pop, Push, Pop