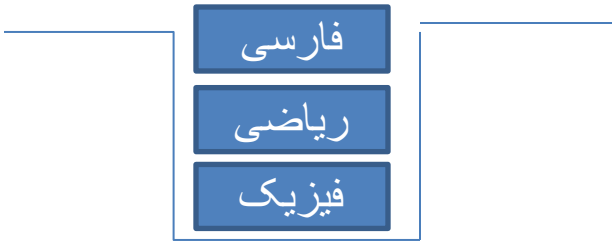
فایل سوم درس ساختمان های داده

- فایل آماده شده برای جلسه معرفی پشته و کاربردهای آن و صرفا برای کلاس آنلاین در این ترم می باشد.

همواره سلامت و موفق باشید- اسبقی

پشته (stack)

برای درک بهتر ساختار پشته، یک کارتن خالی بصورت زیر را در نظر بگیرید:



ساختار پشته با رفتار(LAST IN FIRST OUT)(LIFO): خروج به ترتیب عکس ورود

آیا این ساختار برای پیاده سازی دسترسی های عادلانه مناسب است؟

پشته (stack)

پشته بعنوان یک ADT:

- تعریف:

مجموعه ای مرتب از عناصر بطوریکه دسترسی به عناصر آن، فقط از یک طرف پشته بعنوان بالای پشته(top) انجام می شود.

- عملیات:

-تعریف پشته خالی

-درج عنصر در پشته(push)- در بالای پشته

-حذف عنصر از پشته(pop)- از بالای پشته

-بازیابی عنصر از پشته(Top)- از بالای پشته

پشته (stack)

- پیاده سازی پشته:

❖ پیاده سازی خطی پشته

✓ با استفاده از آرایه

❖ پیاده سازی غیرخطی پشته

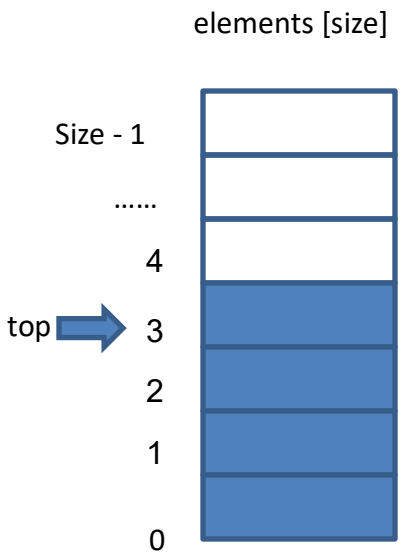
✓ با استفاده از لیست پیوندی

پیاده سازی خطی پشته با استفاده از آرایه

- مولفه های مورد نیاز برای پیاده سازی پشته:

✓ آرایه خطی

✓ متغیر top برای نگهداری محل بالای پشته



تعریف پشته خالی

- برای تعریف پشته از ساختمان(struct) استفاده می کنیم، چرا؟ از ساختارها و انواع داده ای دیگر نیز می توان استفاده نمود؟

تعریف ساختمان:

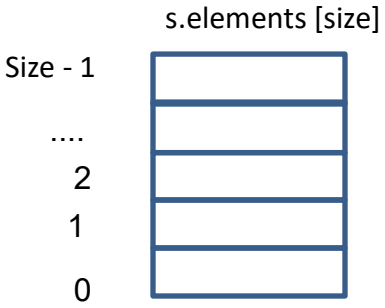
```
struct   نام{
بدنه ساختمان
};
 struct  stack{
 int  elements[size];
 int  top;};
```

- تعریف پشته خالی:

```
stack   s;
s.top = -1;
```

مثال

- ارایه چندین دستور push و pop برای روشن شدن مطلب:



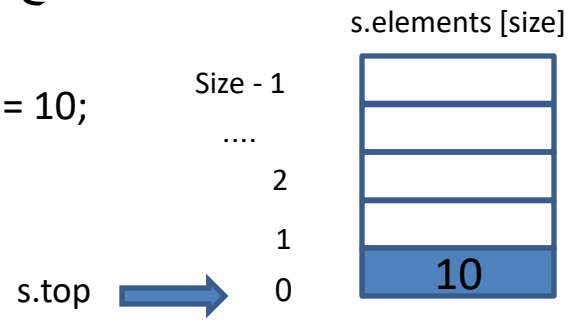
s.top = -1;

1)push(s , 10);

برای اینکار ابتدا باید top یک واحد اضافه شده و سپس مقدار 10 در آن محل آرایه درج شود:

s.top ++;

s.elements[s.top] = 10;

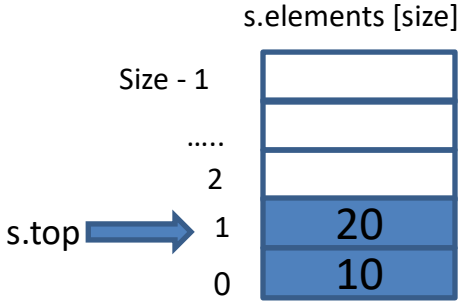


مثال

2)push(s , 20);

برای اینکارنیزابتدا باید top یک واحد اضافه شده و سپس مقدار 20 در آن محل آرایه درج شود:

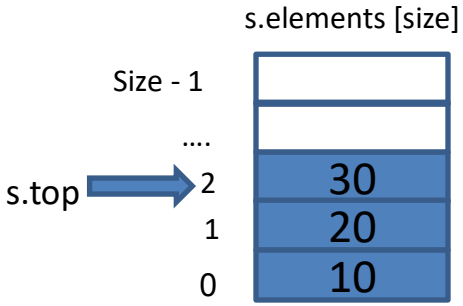
```
s.top ++;
s.elements[s.top] = 20;
```



3)push(s , 30);

برای اینکار ابتدا باید top یک واحد اضافه شده و سپس مقدار 30 در آن محل آرایه درج شود:

```
s.top ++;
s.elements[s.top] = 30;
```



مثال

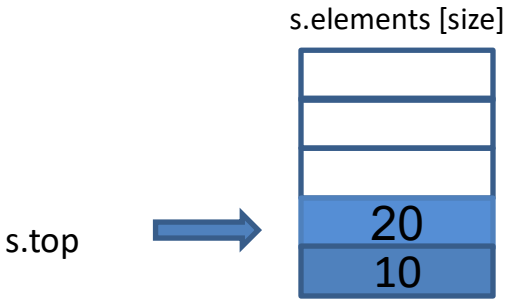
4)pop(s);

برای حذف یک مقدار از پشته،ابتدا باید مقدار مورد نظر در محلی که top به آن اشاره دارد، خوانده شده و سپس از top یک واحد کسر گردد.

X= s.elements[s.top];

s.top- -;

X =30

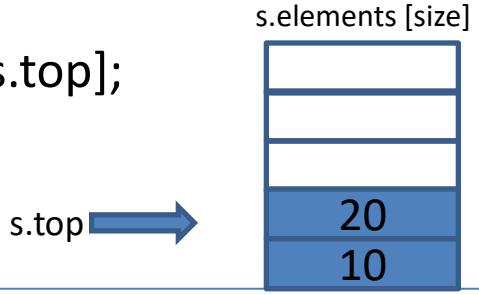


5)Top(s);

برای بازیابی یک مقدار از پشته، باید مقدار مورد نظر در محلی که top به آن اشاره دارد، خوانده شود و نیازی به دستکاری متغیر top وجود ندارد.

X= s.elements[s.top];

X = 20



مثال

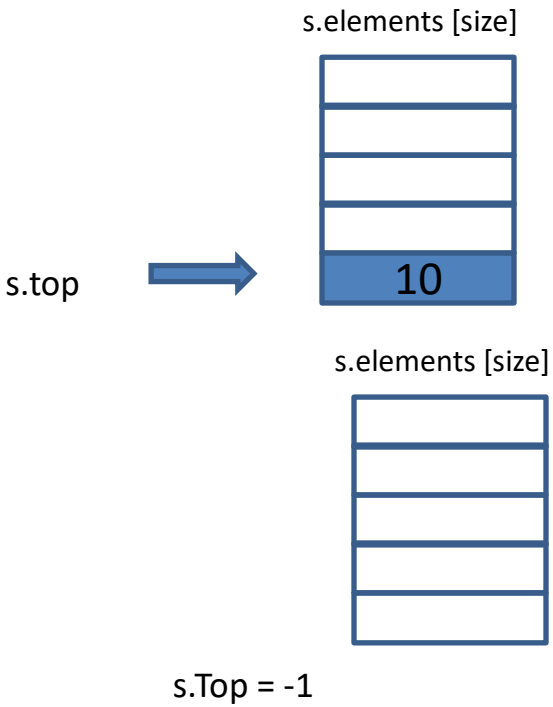
6)pop(s);

برای بازیابی یک مقدار از پشته، باید مقدار مورد نظر در محلی که top به آن اشاره دارد، خوانده شود و نیازی به دستکاری متغیر top وجود ندارد.

X= s.elements[s.top];

s.top- -;

X =20



7)pop(s);

X= s.elements[s.top];

s.top- -;

X =10

پیاده سازی تابع push

```
void  push(stack  *s , int x)
{
   if(s-> top ==size -1)
    { cout<<“stack  overflow”;
       exit(1);}
   else{
       s->top++;
       s->elements[s->top]=x;}
}
```

اگر استک پراست؟

s.elements [++(s->top)]=x;

پیاده سازی تابع pop

```
int  pop(stack  *s )
{
   if(s-> top ==-1)
    { cout<<“stack  underflow”;
       exit(1);}
   else{
        int  x = s->elements[s->top];
        s->top - -;
        return  x;}
}
```

اگر استک خالی است؟

return(s.elements [(s->top)--]);

پیاده سازی تابع Top

```
int  Top(stack  *s )
{
   if(s-> top ==-1)
    { cout<<“stack  underflow”;
      exit(1);}
   else
       return  (x = s->elements[s->top]);
}
```



اگر استک خالی است؟

کاربردهای پشته

دو کاربرد اصلی و پراستفاده پشته عبارتند از:

- فراخوانی توابع
- ارزیابی عبارات

کاربرد پشته در فراخوانی توابع

- مثال زیر را در نظر بگیرید:

```
void   f1();
int    f2(int);
void   f3( int , int);

  int  main(){
    out<<“hello”<<endl;
    f1();
    getch();
  return 0;}
```

```
void f1(){
 int  y, x=10;
 y = f2(x);}
```

```
int f2(int  a){
 int  b= a+2*a;
 f3(a,b);
 return b;}
```

```
void f3(int  c , int d)
{
 cout<< c*d;
}
```

کاربرد پشته در فراخوانی توابع

- ایجاد رکورد فعالیت(Activation Record) برای توابع با هر عمل فراخوانی
- رکورد فعالیت شامل:
  - -آدرس های برگشت
  - -متغیرهای محلی
  - -پارامتر ها
  - -ثبات ها
  - ....
- نگهداری رکوردهای فعالیت در پشته به علت خاصیت پشته
- رکورد فعالیت بالای پشته مربوط به تابع در حال اجرا

کاربرد پشته در فراخوانی توابع

```
void   f1();
int    f2(int);
void   f3( int , int);

int  main(){

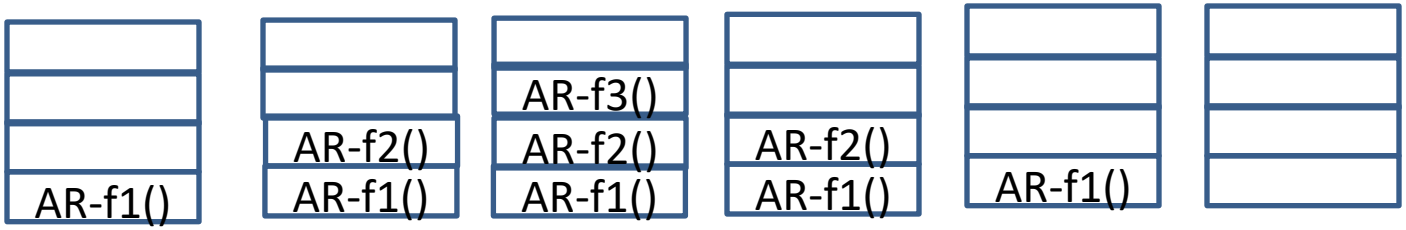
out<<“hello”<<endl;
  f1();
  getch();
return 0;}
```

```
void f1(){
 int  y, x=10;
 y = f2(x);}
```

```
int f2(int  a){
 int  b= a+2*a;
 f3(a,b);
 return b;}
```

```
void f3(int  c , int d)
{
 cout<< c*d;
}
```

1 2 3 4 5 6



کاربرد پشته در ارزیابی عبارات

- ✓ عبارات به سه صورت زیر قابل تعریف می باشند:

- عبارت پیشوند(prefix) +A*BC
- عبارت میانوند(infix) A+B*C
- عبارت پسوند(postfix) ABC*+

- ✓ عبارات پسوند و پیشوند برخلاف ظاهرشان به سادگی قابل استفاده می باشند.
- ✓ تبدیل عبارات میانوند به عبارات پسوند و پیشوند توسط کامپایلرها

تبدیل عبارت میانوند به عبارت پسوند با استفاده از پشته

- ساختار عبارت میانوند:

-عملوند(operand) ABC

-عملگر(operator) + - */

-پرانتز باز

-پرانتز بسته

- ساختار عبارات پسوند و پیشوند:

-عملوند

-عملگر

الگوریتم تبدیل عبارت میانوند به عبارت پسوند با استفاده از پشته

- ایجاد یک پشته خالی
- پیمایش تک تک مولفه های عبارت میانوند به ترتیب:

✓ اگر مولفه عبارت میانوند عملوند باشد:

دراینصورت در عبارت پسوند نهایی تعریف می شود.

✓ اگر مولفه، عملگر باشد:

دراینصورت در پشته push می گردد، البته به شرطی که اولویت آن از اولویت مولفه بالای داخل پشته بیشتر باشد. درصورتیکه دارای اولویت کمتر یا مساوی باشد، عملگر بالای پشته pop شده و سپس با درنظر گرفتن شرط قبلی، عملگر جدید push میشود.

✓ اگر مولفه، پرانتز باز باشد:

در این حالت، در پشته push می شود.

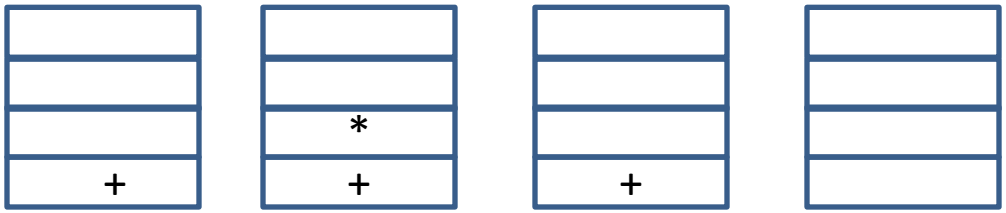
✓ و اگر مولفه عبارت میانوند، پرانتز بسته باشد:

در اینصورت تا اولین پرانتز باز از پشته pop می گردد.

الگوریتم تبدیل عبارت میانوند به عبارت پسوند با استفاده از پشته

- مثال: عبارت میانوند زیر را در نظر بگیرید:

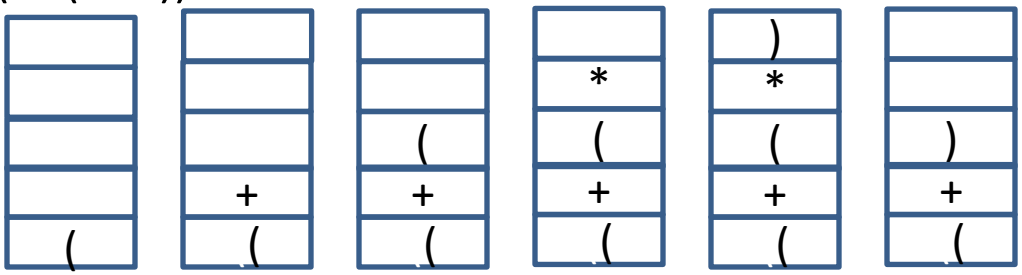
A+B*C



1-عبارت پسوند:A
2-عبارت پسوند:AB
3-عبارت پسوند:ABC
4- عبارت پسوند نهایی:ABC*+

- مثال: عبارت میانوند زیر را در نظر بگیرید:

(A+(B*C))

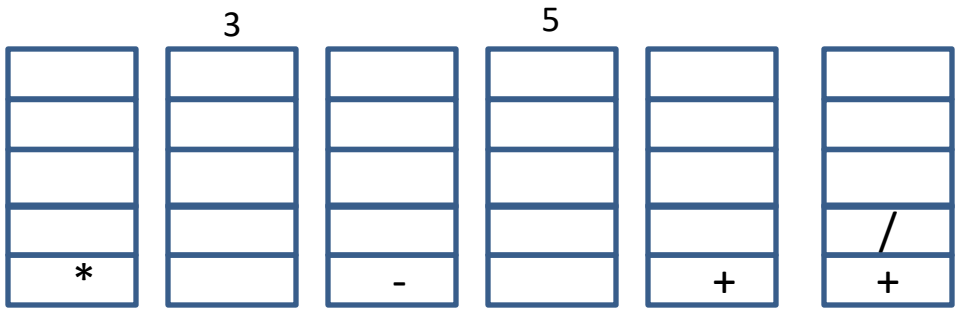


1-عبارت پسوند:A
2-عبارت پسوند:AB
3-عبارت پسوند:ABC
4- عبارت پسوند نهایی:ABC*+

الگوریتم تبدیل عبارت میانوند به عبارت پسوند با استفاده از پشته

- مثال: عبارت میانوند زیر را در نظر بگیرید:

A*B-C+D/K



1-عبارت پسوند:A

2-عبارت پسوند:AB

3-عبارت پسوند:AB*

4- عبارت پسوند:AB*C

5- عبارت پسوند:AB*C-

6- عبارت پسوند:AB*C-D

7- عبارت پسوند:AB*C-Dk

8- عبارت پسوند:AB*C-DK/+

الگوریتم تبدیل عبارت میانوند به عبارت پسوند با استفاده از پشته

- مثال: عبارت میانوند زیر را در نظر بگیرید:

A/B*C+D-K

1-عبارت پسوند:A

2-عبارت پسوند:AB

3-عبارت پسوند:AB/

4- عبارت پسوند:AB/C

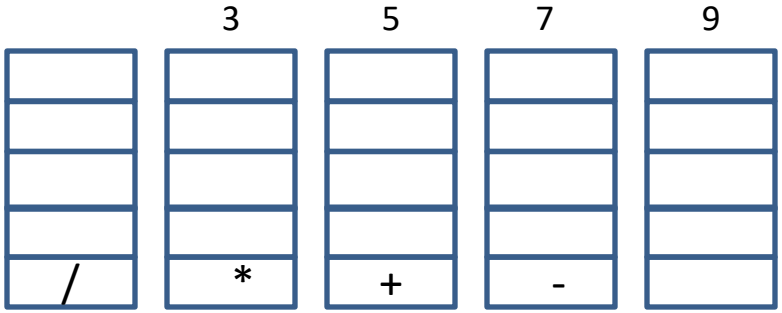
5- عبارت پسوند:AB/C*

6- عبارت پسوند:AB/C*D

7- عبارت پسوند:AB/C*D+

8- عبارت پسوند:AB/C*D+K

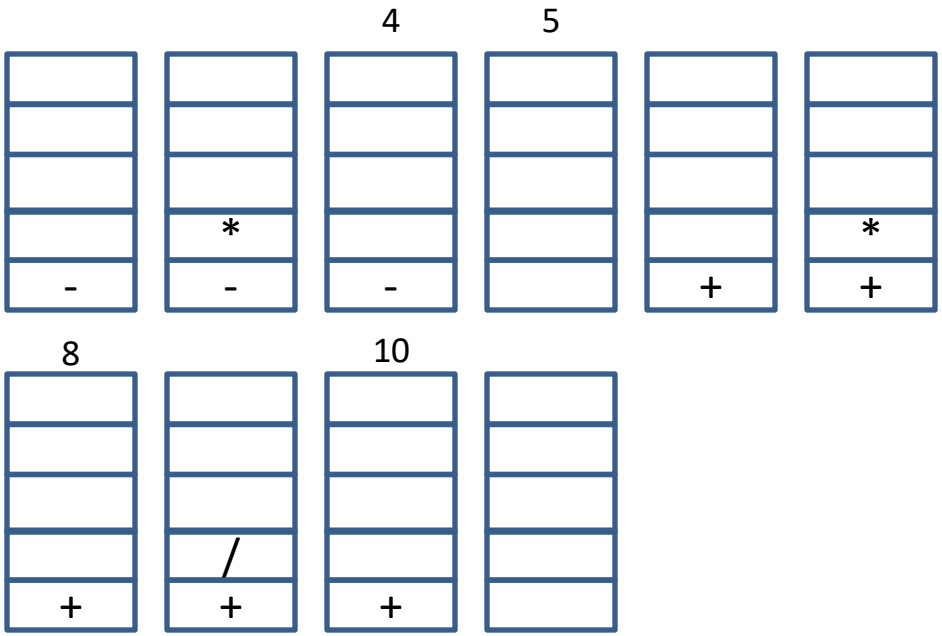
9- عبارت پسوند:AB/C*D+K-



الگوریتم تبدیل عبارت میانوند به عبارت پسوند با استفاده از پشته

- مثال: عبارت میانوند زیر را در نظر بگیرید:

A-B*C+D*K/P



1-عبارت پسوند:A
2-عبارت پسوند:AB
3-عبارت پسوند:ABC
4- عبارت پسوند:ABC*
5- عبارت پسوند:ABC*-
6- عبارت پسوند:ABC*-D
7- عبارت پسوند:ABC*-DK
8- عبارت پسوند:ABC*-DK*
9- عبارت پسوند:ABC*-DK*P
10- عبارت پسوند:ABC*-DK*P/
11- عبارت پسوند:ABC*-DK*P/+

الگوریتم ارزیابی عبارت پسوند بصورت عبارت میانوند با استفاده از پشته

- ایجاد یک پشته خالی
- پیمایش تک تک مولفه های عبارت پسوند به ترتیب:

✓ اگر مولفه عبارت پسوند عملوند باشد:

دراینصورت در پشته push می شود.

✓ اگر مولفه عبارت پسوند، عملگر باشد:

دراینصورت این عملگر بر روی دو عملوند بالای پشته اعمال شده و نتیجه مجددا بعنوان یک عملوند در پشته push می گردد.

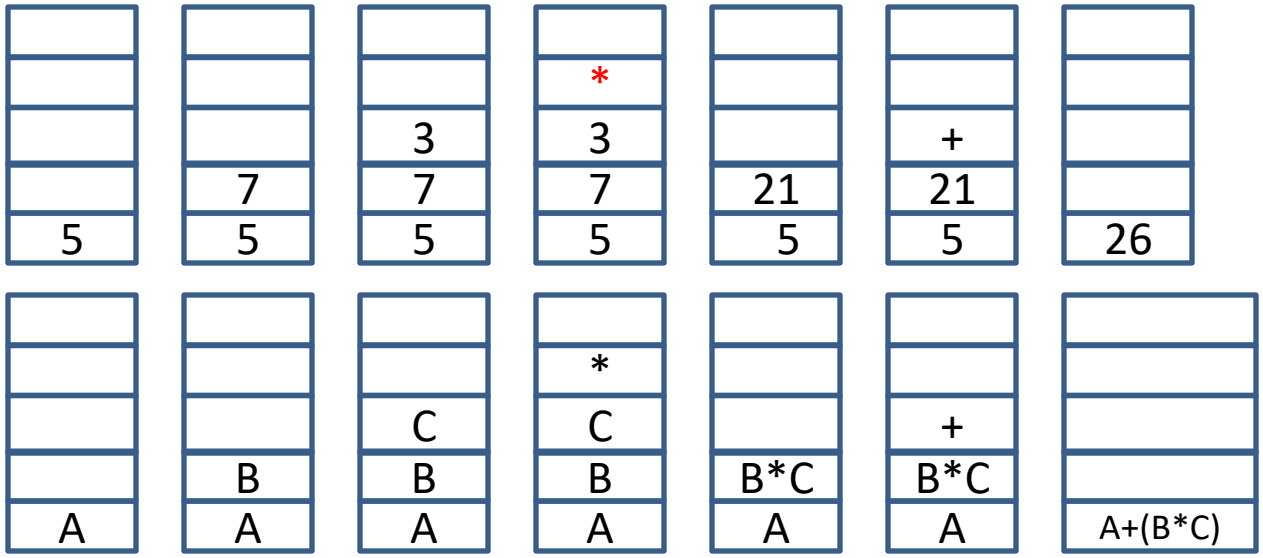
- نتیجه نهایی بصورت عملوند در پشته قرار می گیرد.

الگوریتم ارزیابی عبارت پسوند بصورت عبارت میانوند با استفاده از پشته

- مثال: عبارت پسوند زیر را در نظر بگیرید:

ABC*+

573*+



تمرین

- عبارات میانوند زیر را با استفاده از پشته به عبارت پسوند تبدیل نمایید.(با نمایش مراحل)

الف)A+B/C-D*K+P

ب)A*B*C+D/K-P

ج)A/B*(C+D*K)/P

- عبارات پسوند بدست آمده را مجددا با استفاده از پشته بصورت عبارت میانوند اولیه ارزیابی نمایید.(با نمایش مراحل)