ساختمان داده‌ها

دکتر امین گلزاری اسکوئی

a.golzari@azaruniv.ac.ir
a.golzari@tabrizu.ac.ir
https://github.com/Amin-Golzari-Oskouei

دانشگاه شهید مدنی آذربایجان

پاییز ۱۴۰۲

فصل ۴

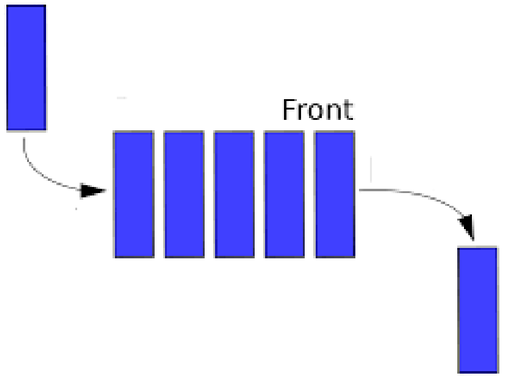
صف و پشته

مطالب این فصل

- تعریف صف و پشته
- خواص صف و پشته و کاربردهای آن

صف (queue)

صف، ساختمان داده ای است که عمل حذف از ابتدای آن و درج به انتهای آن انجام می‌شود.



صف را لیست FIFO (First In First Out) می نامند، زیرا اولین عنصر وارد شده به صف، اولین عنصری است که خارج می‌شود.

برای نمایش صف، از آرایه یک بعدی queue[0..n-1] و دو متغیر front و rear استفاده می‌شود.

متغیر front : یکی کمتر از مقدار محل عنصر اول صف

متغیر rear : محل آخرین عنصر صف

مثال

یک صف با 4 خانه که در ابتدا خالی است مفروض می‌باشد. (q[0..3])

0	1	2	3
A			
A	B		
A	B	C	
	B	C	
		C	

front = -1 , rear = -1

front = -1 , rear = 0 درج A

front = -1 , rear = 1 درج B

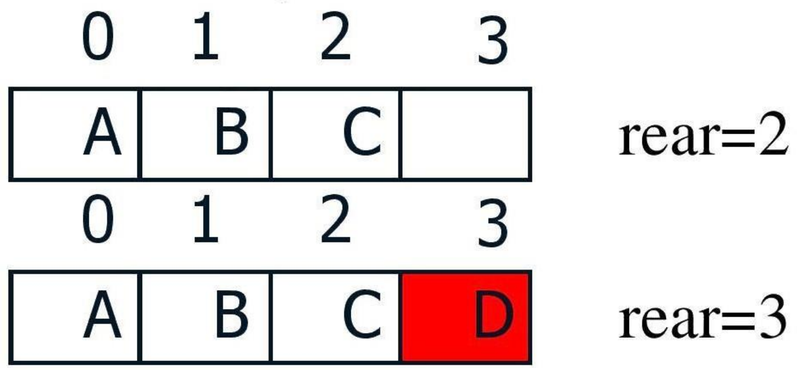
front = -1 , rear = 2 درج C

front = 0 , rear = 2 حذف A

front = 1 , rear = 2 حذف B

درج در صف

```
addq (rear ,  item)
{
  if (rear == n-1)
        {
          queue-full( );
          return;
         }
    rear= rear+1;
    queue[rear] = item;
}
```

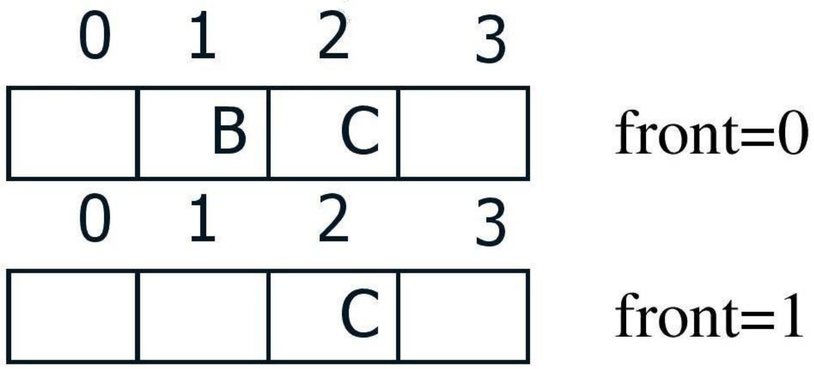


تذکر: در صف پر مقدار rear برابر n-1 می باشد.

تذکر: در صف خالی مقدار front با rear برابر است.

حذف عنصر از صف

```
delq (front , rear)
{
   if (front == rear)
         return queue-empty( );
   return   q[++front] ;
}
```



مشکل نمایش ترتیبی صف

نمایش ترتیبی صف، دارای نقاط ابهامی است. با ورود و خروج داده ها ، صف بتدریج بطرف راست تغییر مکان می دهد. به نحوی که rear برابر n-1 می شود و به نظر می رسد که صف پر است.

در این حالت ، queue-full باید تمام صف را به سمت چپ شیفت دهد و اولین عنصر دوباره در موقعیت queue[0] قرار گرفته و front برابر 1- شود. rear نیز باید تنظیم شود. ولی شیفت عملی زمانبر است.

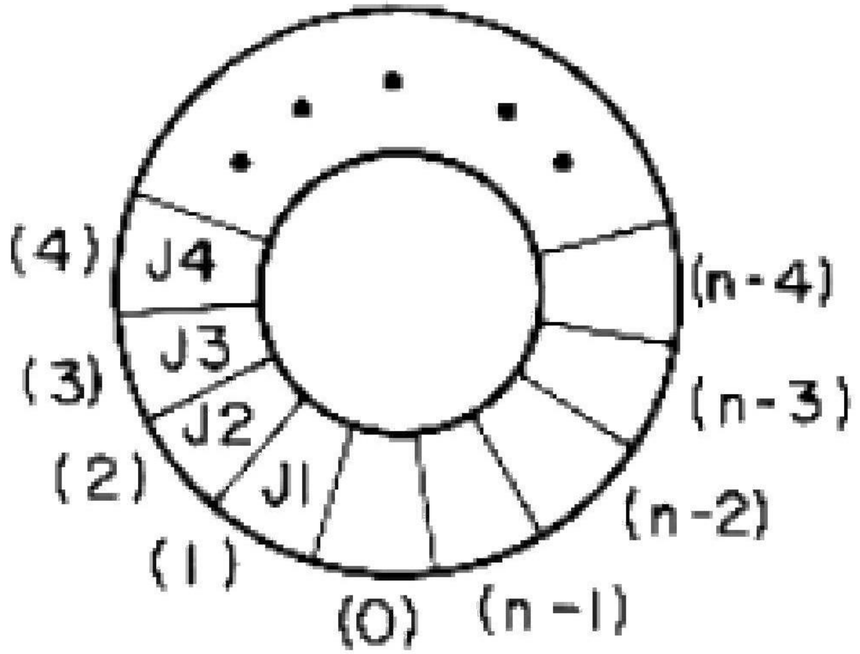
صف حلقوی

در صف حلقوی اندیس front به یک موقعیت عقب تر (در خلاف حرکت عقربه های ساعت) از اولین عنصر موجود در صف اشاره می کند. rear به انتهای فعلی صف اشاره می کند.

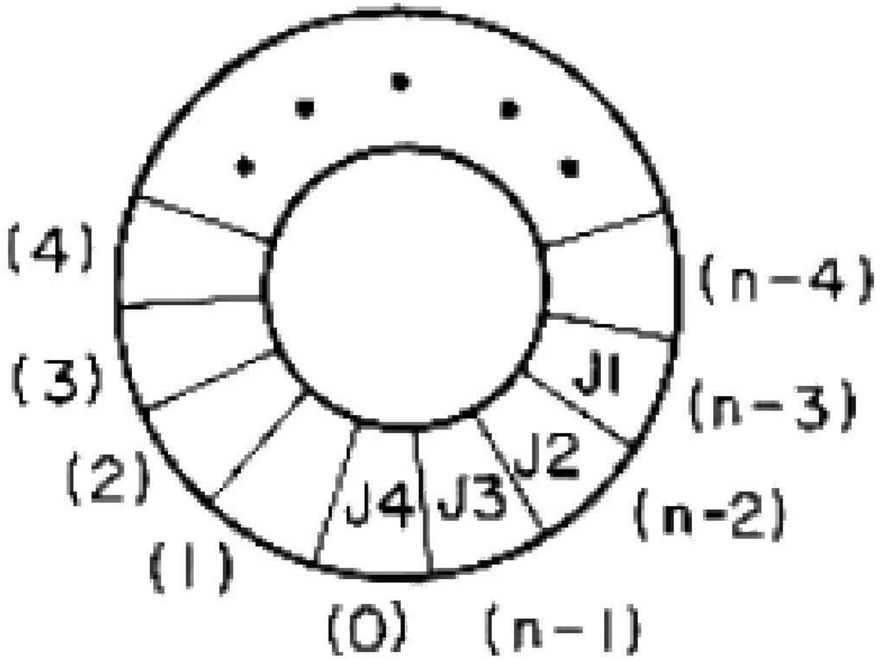
در یک صف حلقوی به اندازه n، حداکثر n-1عنصر می تواند، قرار گیرد.

اگر از آن یک خانه نیز استفاده شود، front=rear می شود و نمی توانیم یک صف پر و خالی را از هم تشخیص دهیم.

مثال



front =0; rear = 4



front = n-4, rear = 0

درج در صف حلقوی

```
addq (front , rear , item)
{
    rear = (rear+1) % n;
     if ( rear == front )
         {
            queue-full(rear);
            return;
         }
    queue[rear] = item ;
}
```

حذف از صف حلقوی

```
deleteq (front , rear)
{
  if (front == rear)
      return  queue-empty( );
  front = (front+1)  %  n ;
   return   queue[front] ;
}
```

اگر front=rear باشد، صف خالی خواهد بود.

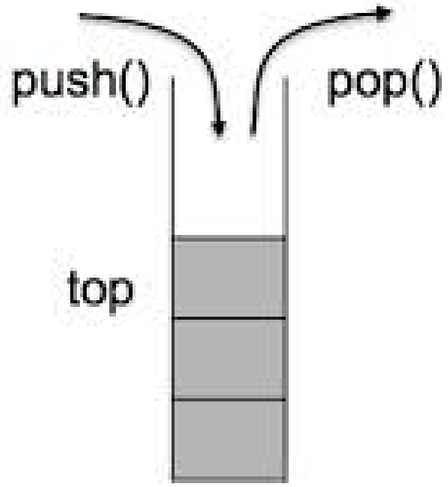
پیاده سازی توابع queue-full و queue-empty بسته به کاربردهای خاص دارد.

پشته (stack)

پشته ، ساختمان داده ای است که حذف و اضافه از بالای آن انجام می‌شود.

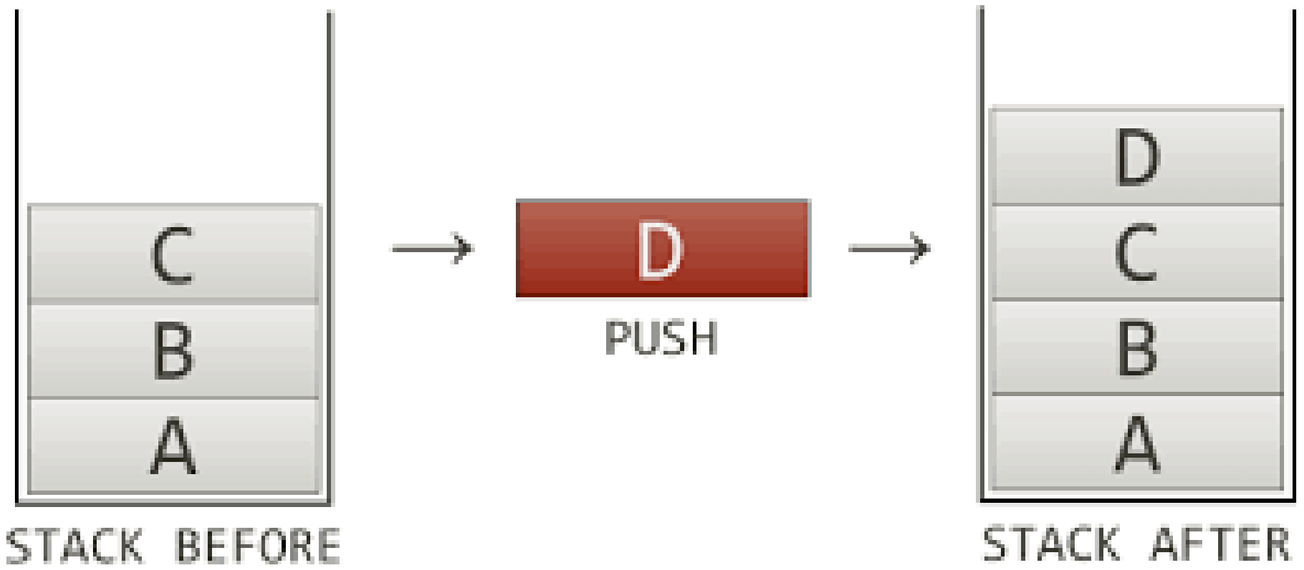
پشته را LIFO می‌نامند، چون آخرین عنصر وارد شده در آن، اولین عنصری است که از آن برداشته می‌شود.

top : مشخص کننده عنصر بالایی پشته



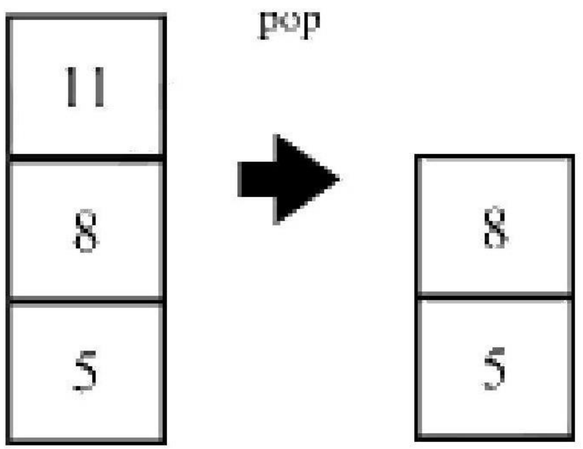
درج در پشته

```
push (top , item)
{
    if ( top >= max-1 )
     {
      stack-full( );
      return;
     }
    top=top+1;
    stack[top] = item;
}
```



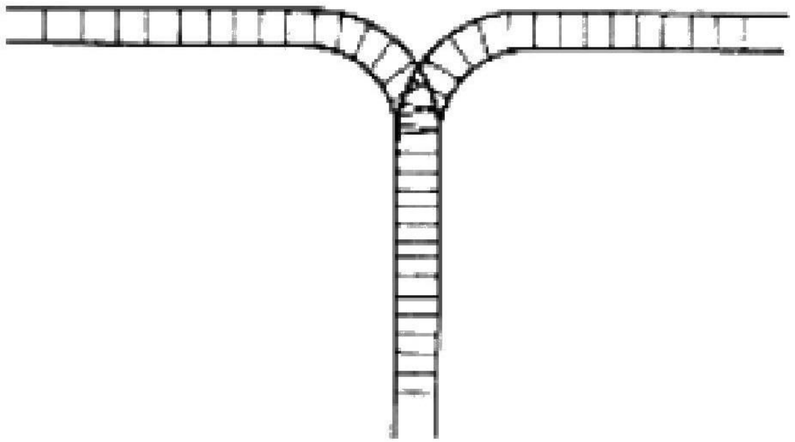
حذف از پشته

```
pop ( top )
{
    if ( top == -1 )
            return  stack-empty( );
    return
            stack[top--];
}
```



جابه‌جایی قطارها

در ایستگاه‌های قطار برای جابه‌جا کردن قطارها از یک ریل اضافی(S) استفاده می کنند که مثل پشته عمل می کند. یعنی اولین قطار که وارد این ریل می شود آخرین قطاری است که از آن خارج می شود. فرض کنید در ریل ورودی دنباله‌ای از قطارها با شماره های < 1 , 2 , 3 > پشت سر هم قرار دارند. (۱ در ابتدای ردیف است). اگر مراحل زیر انجام شود، خروجی چه خواهد بود؟



۱- قطار ۱ وارد S می شود.

۲- قطار ۲ وارد S می شود.

۳- قطار ۲ از S خارج شده و وارد ریل خروجی می شود.

۴- قطار ۳ وارد S می شود.

۵- قطار ۳ از S خارج شده و به ریل خروجی می رود.

۶- قطار ۱ از S خارج شده و به ریل خروجی می رود.

حل:

ترتیب خروج قطارها : < 2 , 3 , 1 >

مثال

در مسئله قطارها ، آیا ترتیب خروجی <2 , 1 , 3> ممکن است؟

حل:

ابتدا 1 و2 و 3 وارد s شده و سپس 3 را خارج می کنیم. ولی بعد از آن نمی توان 1 را خارج کرد، چون قبل از آن باید 2 را خارج کرد.

تذکر:

اعداد <1,2,3> دارای ۶ جایگشت هستند، ولی جایگشت <3,1,2> در خروجی ممکن نیست.

< 1 , 2 , 3 >
< 1 , 3 , 2 >
< 2 , 1 , 3 >
< 2 ,3 , 1 >
< 3 , 2 , 1 >

مثال

اعداد 1 تا 8 به ترتیب در پشته قرار دارند (۸ در بالای پشته). عمل pop و push به صورت زیر تعریف شده اند.

Push : اولین عدد ورودی را برداشته و در بالای پشته قرار می دهد.

Pop : عدد بالای پشته را برداشته و در انتهای دنباله خروجی می نویسد.

با ترکیب مناسبی از ۸ عدد push و ۸ عدد pop می توان جایگشتی از اعداد ۱تا ۸را در خروجی تولید کرد که به آن جایگشت قابل قبول می گوییم. آیا جایگشت زیر قابل قبول است؟

$< 4,3,7,8,6,2,5,1 >$

حل:

push(1) , push(2) , push(3) , push(4) , pop(4) , pop(3) , push(5) , push(6) , push(7) , pop(7) , push(8) , pop(8) , pop(6) , ???

بعد از بیرون آوردن ۶ ، نمی توان ۲ را خارج کرد ، چون زیر ۵ مانده است.

راه سریع: برای آنکه یک دنباله خروجی قابل قبول باشد ،از انتها به سمت ابتدا حرکت کرده و برای هر عدد x ، اعداد کوچکتر از x که بعد از آن قرار دارند، باید یک دنباله نزولی باشند. در این دنباله ، اعداد بعد از 6 یعنی <2,5,1>، یک دنباله نزولی نمی باشند.

نگارش های مختلف عبارت ریاضی

یک عبارت از عملوندها و عملگرها ساخته شده که می‌تواند به سه شکل نمایش داده شود:

۱- prefix (پیشوندی) * A B

۲- infix (میانوندی) A * B

۳- postfix (پسوندی) A B *

اولویت بین عملگرهایی دودویی:

۱- توان

۲- ضرب و تقسیم

۳- جمع و تفریق

تبدیل infix به postfix

مثال :

((A + B) * D) ^ (E – F)

(**AB+** *D) ^ (E – F)

AB+D* ^ (E – F)

AB+D* ^ **EF-**

AB+D*EF-^

تذکر: در نگارش prefix و postfix یک عبارت، نیازی به پرانتز گذاری نیست و ترتیب عملگرها از جایگاه آن‌ها مشخص می شود.

تبدیل infix به prefix

مثال:

((A + B) * D) ^ (E – F)

(+**AB** * D) ^ (E – F)

***+ABD** ^ (E – F)

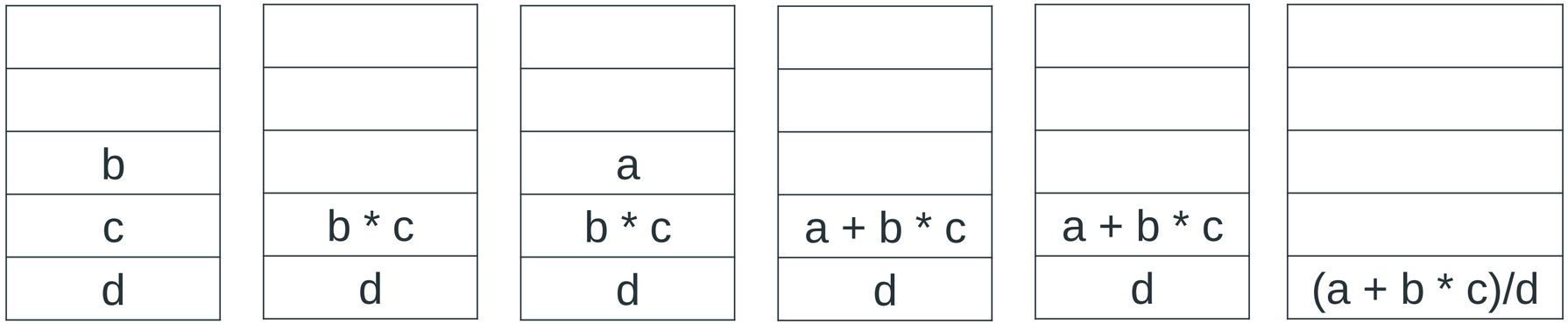
***+ABD ^ -EF**

^*+ABD-EF

تبدیل prefix به infix با استفاده از پشته

مثال :

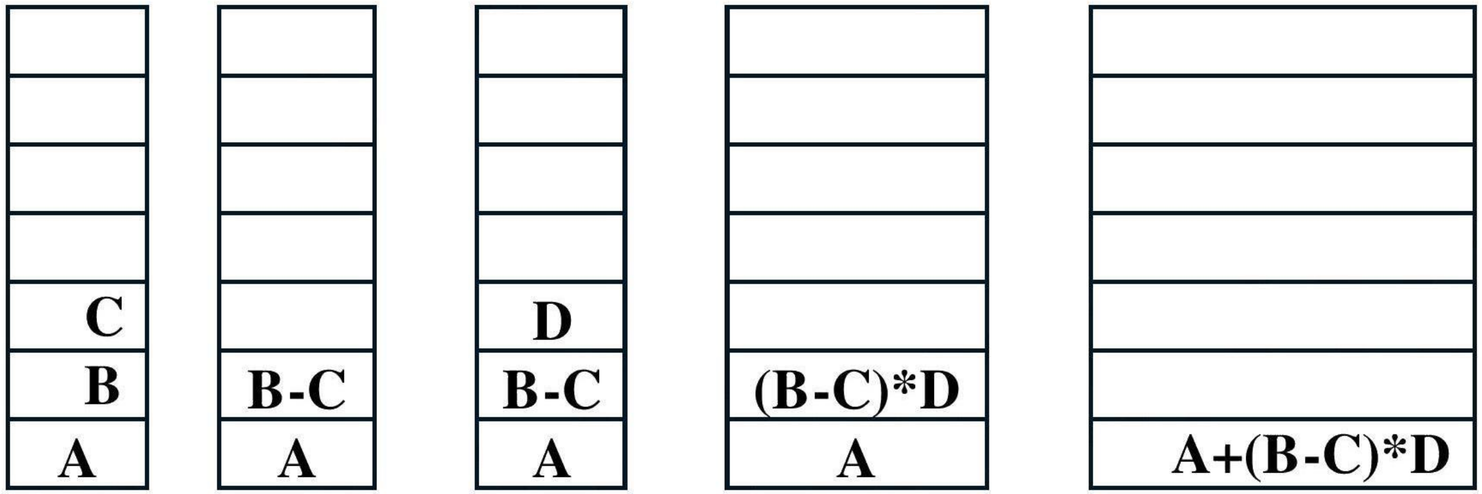
/+ a * b c d



تبدیل postfix به infix با استفاده از پشته

مثال :

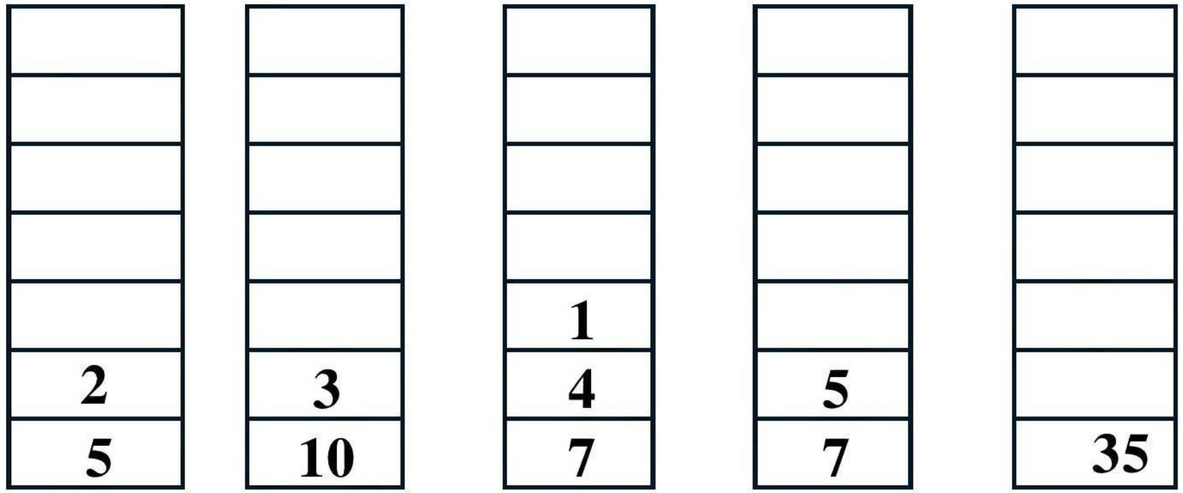
A B C – D * +



مثال

حاصل عبارت زیر را مشخص کنید.

5 , 2 , * , 3 , - , 4 , 1 , + , *



تمرین

فرم **postfix** را مشخص کنید.

Infix Expression	Prefix Expression
A +B-C	-+ABC
(A+B) *(C+D)	*+AB+CD
A/B*C-D+E/F/(G+H)	+-*/ABCD//EF+GH
((A + B) * C - (D-E))*(F+G)	*-*+ABC-DE +FG
A-B/(C *D/E)	-A/B/*CDE

PYTHON TIME

پیاده‌سازی صف

```
class Queue:

  def __init__(self, k):
    self.k = k
    self.queue = [None] * k
    self.front = -1
    self.rear = -1
```

```
def display(self):
    if(self.front == -1):
      print('empty')
    for i in range(self.front, self.rear+1):
        print(self.queue[i], end=' ')
    print()
```

در این پیاده سازی فرض بر این است که front عنصر اول و rear عنصر آخر است

درج در صف – حذف از صف

```
def enqueue(self, data):
  if((self.rear + 1) == self.k):
    print('full')
  elif (self.front == -1):
    self.front = 0
    self.rear = 0
    self.queue[self.rear] = data
  else:
    self.rear += 1
    self.queue[self.rear] = data
```

```
def dequeue(self):
    if (self.front == -1):
      print('empty')
    elif (self.front == self.rear):
      t = self.queue[self.front]
      self.front = -1
      self.rear = -1
      return t
    else:
      t = self.queue[self.front]
      self.front += 1
      return t
```

درج در صف چرخشی

```
def enqueue(data):
    if((rear + 1) % k  == front):
      print('full')
    elif (front == -1):
      front = 0
      rear = 0
      queue[rear] = data
    else:
      rear = (rear +1) % k
      queue[rear] = data
```

حذف از صف چرخشی

```
def dequeue( ):
      if (front == -1):
        print('empty')
        return
      elif (front == rear):
        t = queue[front]
        front = -1
        rear = -1
        return t
      else:
        t = queue[front]
        front = (front + 1) % k
        return t
```

پیاده‌سازی پشته

```
class Stack( ):
    def __init__(self, limit=10):
        self.stack = []
        self.limit = limit
```

```
    def peek(self):
        if len(self.stack) <= 0:
            return -1
        else:
            return self.stack[len(self.stack) - 1]
```

پیاده‌سازی پشته

```
class Stack( ):
    def __init__(self, limit=10):
        self.stack = []
        self.limit = limit
```

```
    def peek(self):
        if len(self.stack) <= 0:
            return -1
        else:
            return self.stack[len(self.stack) - 1]
```

درج در پشته – حذف از پشته

```
def push(self, data):
    if len(self.stack) >= self.limit:
        return -1
    else:
        self.stack.append(data)
```

```
def pop(self):
    if len(self.stack) <= 0:
        return -1
    else:
        return self.stack.pop()
```

تبدیل از مبنای ده‌دهی به دودویی

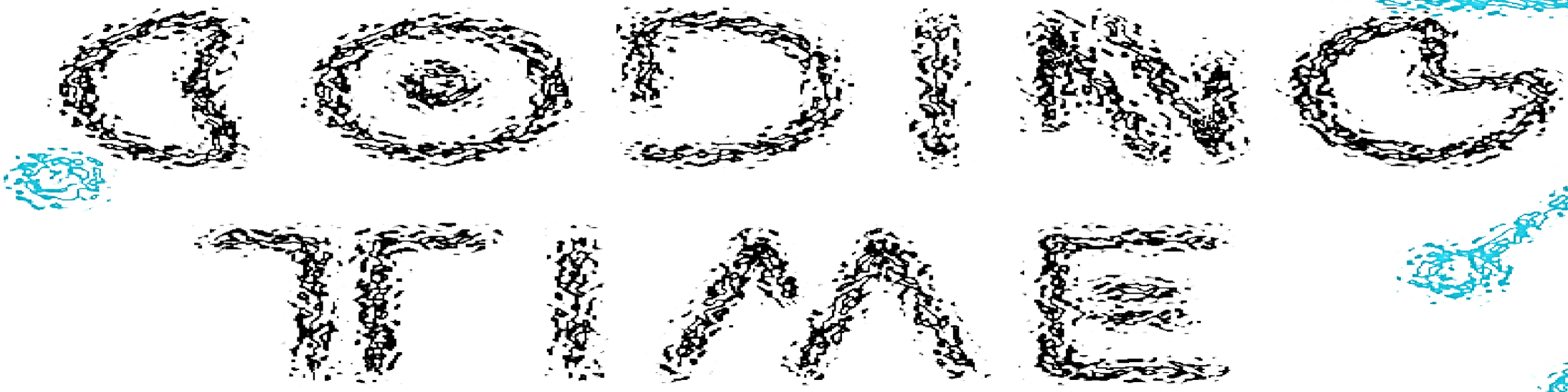
```
def d2b(n):
    s = Stack()

    while n > 0:
        r = n % 2
        s.push(r)
        n = n // 2

    b = ""
    while not s.is_empty():
        b = b + str(s.pop())
    return b
```

معکوس‌کردن لیست به کمک پشته

```
def reverse(lst):
    s = Stack()
    for e in lst:
        s.push(e)

    for i in range(len(lst)):
        lst[i] = s.pop()
```

معکوس‌کردن محتویات پشته

```
def reverse_stack(S):
    s1 = Stack()
    s2 = Stack()
    while not S.is_empty():
        s1.push(S.pop())

    while not s1.is_empty():
        s2.push(s1.pop())

    while not s2.is_empty():
        S.push(s2.pop())
```


GOD IS
TIME

تشکر

سوال؟

a.golzari@azaruniv.ac.ir
a.golzari@tabrizu.ac.ir
https://github.com/Amin-Golzari-Oskouei