ساختمان داده ها و الگوریتم ها

آرایه و پشته

محمدحسین اولیائی

رئوس مطالب

- تعریف آرایه
- پشته
  - مدیریت فراخوانی توابع
  - تبدیل عبارت میانوندی به پسوندی
  - مسئله Maze
  - ارزشیابی عبارت های پسوندی

آرایه

مجموعه ای از خانه های متوالی حافظه که همگی از یک نوع و دارای یک نام می باشند.

هریک از خانه ها دارای اندیس مشخصی هستند که آنها را از یکدیگر متمایز می کند

مثال :int arr[20]

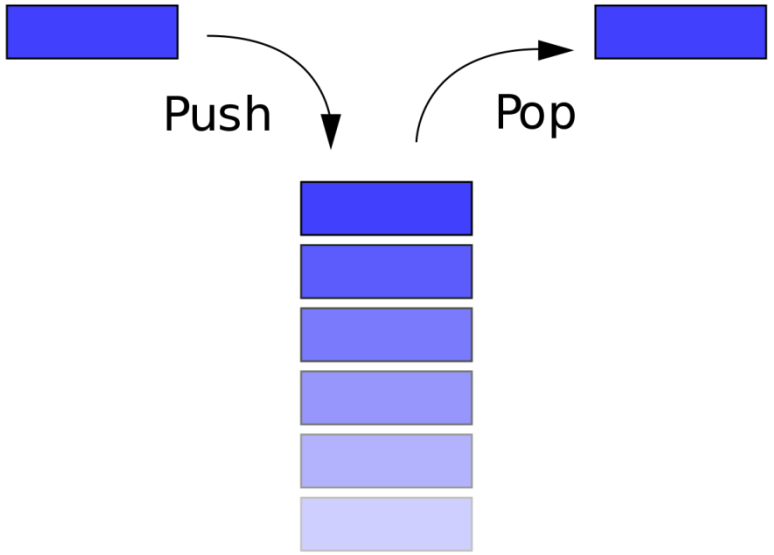
همانطور که می دانید دسترسی به هریک از خانه ها بطور مستقیم می باشد

اعمال متداول

- ایجاد آرایه
- محاسبه تعداد عناصر موجود در آرایه
- درج در ابتدا یا انتهای آرایه
- حذف یک عنصر از آرایه
- جستجوی یک مقدار
- مرتب سازی لیست
- یافتن عناصر مشترک بین دو آرایه
- و...

پشته(Stack)

- لیستی از عناصر که در آن هم اضافه کردن و هم حذف کردن از یک سوی لیست که top نامیده می شود، انجام می گیرد
- بعبارت دیگر، آخرین عنصر ورودی، اولین عنصر خروجی می باشد(LIFO)



پشته(Stack)

- چهار عمل اصلی در پشته:
  - Push: اضافه کردن عنصر به بالای پشته
  - Pop: حذف از بالای پشته
  - بررسی پر بودن
  - بررسی خالی بودن

پیاده سازی با آرایه

```
class stack{
	private:
		char *s;
		int top;
		int maxsize;
	public:
		stack(int m);
		int IsFull();
		int IsEmpty();
		int ElementNumbers();
		char pop();
		void push(char );
		void show();
};
```

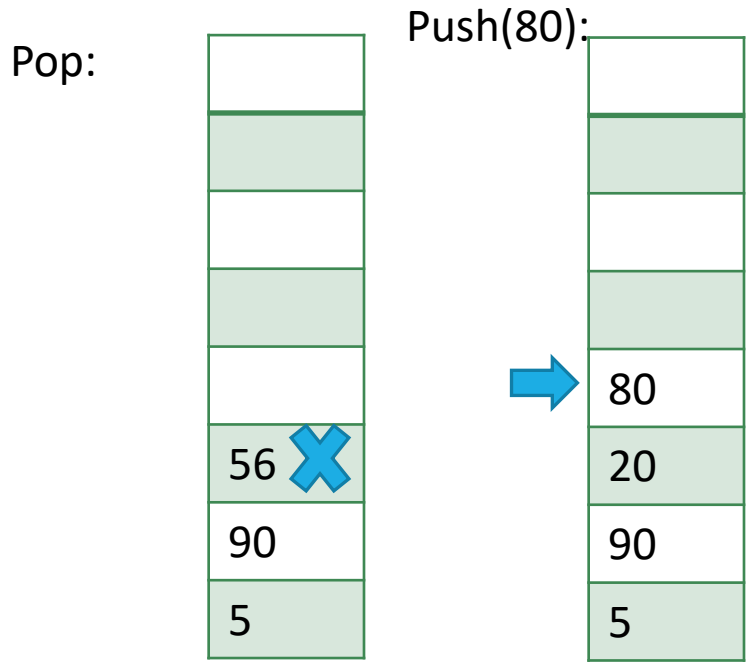
پیاده سازی با آرایه

```
stack::stack(int m)
{
	s=new char[m];
	maxsize=m;
	top=-1;
}
int stack::IsFull()
{
	if (top==maxsize-1)
		return 1;
	return 0;
}
```

```
int stack::IsEmpty()
{
	if(top==-1)
		return 1;
	return 0;
}
```

پیاده سازی با آرایه

```
char stack::pop()
{
	if(!IsEmpty())
		return s[top--];
	else
		{
			cout<<"Stack is empty!\n";
			return '\0';
		}
}
void stack::push(char m)
{
	if(!IsFull())
	{
		s[++top]=m;
	}
}
```



کاربردهای پشته

- مدیریت فراخوانی توابع
- تبدیل عبارت میانوندی به پسوندی
- مسئله Maze
- ارزشیابی عبارت های پسوندی

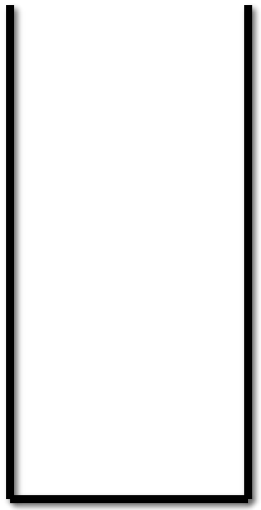
مدیریت فراخوانی توابع

- When a function is called, the run-time system pushes on the stack a frame containing
  - Local variables and return value
  - Program counter, keeping track of the statement being executed
- When a function returns, its frame is popped from the stack and control is passed to the method on top of the stack

مدیریت فراخوانی توابع

```
main(){
   int i=5;
   foo(i);
   i++;
   Cout<< i;}
foo(int j){
    int k;
    k=j+1;
    bar(k);}
bar(int m){
    ….}
```

Calling main

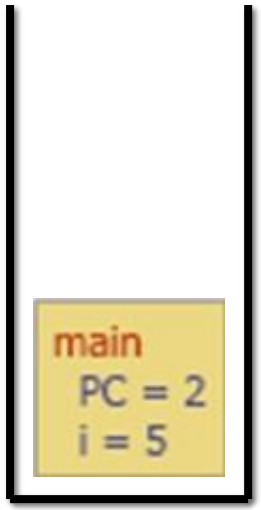


top=-1

مدیریت فراخوانی توابع

```
main(){
   int i=5;
   foo(i);}
foo(int j){
    int k;
    k=j+1;
    bar(k);}
bar(int m){
    ….}
```

Calling foo

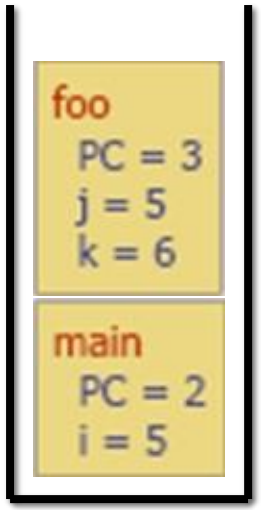


top=0

مدیریت فراخوانی توابع

```
main(){
   int i=5;
   foo(i);}
foo(int j){
    int k;
    k=j+1;
    bar(k);}
bar(int m){
    ….}
```

Calling bar

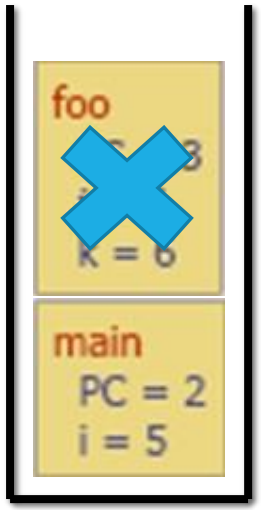


top=1

مدیریت فراخوانی توابع

```
main(){
    int i=5;
    foo(i);}
foo(int j){
     int k;
     k=j+1;
     bar(k);}
bar(int m){
     ….}
```

terminating bar

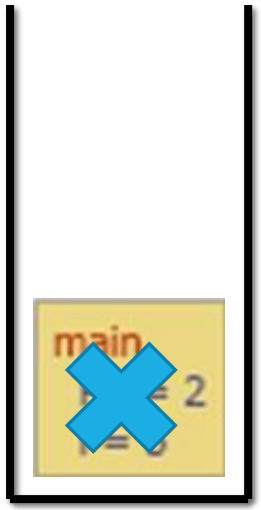


top=0

مدیریت فراخوانی توابع

```
main(){
   int i=5;
   foo(i);}
foo(int j){
    int k;
    k=j+1;
    bar(k);}
bar(int m){
    ….}
```

terminating foo



top=-1

مثال: مدیریت فراخوانی توابع زیر را مرحله به مرحله نشان دهید. حداقل تعداد خانه های این پشته چقدر باید باشد؟

```
f1(){
.
.
.
f2();
.
.}
```

```
f2(){
.
.
.
f3();
.
.}
```

```
f3(){
.
.
.
f4();

}
```

```
f4(){
.
.
.
f5();
}
```

```
f5(){
.
.
.
f6();
...
}
```

```
f6(){
.
.
.
}
```

کاربردهای پشته

- مدیریت فراخوانی توابع
- تبدیل عبارت میانوندی به پسوندی
- مسئله Maze
- ارزشیابی عبارت های پسوندی

تبدیل عبارت میانوندی به پسوندی

- نمایش میانوندی: x+y
- نمایش پیشوندی: +xy
- نمایش پسوندی: xy+

مثال:

$(a+b)*c^2 \rightarrow ab+c\wedge 2*$

Token	Operator	Associativity
()	Function call	Left-to-right
!	Logical not	Right- to-left
* / %	Multiplicative	Left-to-right
+ -	Binary add or subtract	Left-to-right
> >= < <=	Relational	Left-to-right
== !=	Equality	Left-to-right
&&	Logical and	Left-to-right
||	Logical or	Left-to-right
= += -= /= *= %= <<= >>= &= ^= |=	Assignment	Right- to-left

Infix to Postfix Conversion

ALGORITHM

1. Push “(“ onto Stack and Add “)” to the end of Q.
2. Scan Q from left to right and repeat steps 3 to 6 for each element of Q until Stack is empty:
3. If an operand is encountered, add it to P.
4. If a Left parenthesis is encountered, push it on Stack.
5. If an Operator ¤ encountered, then :
    a) Repeatedly pop from Stack and add to P each operator which has the same precedence as or higher precedence than ¤.
    b) Add ¤ to the Stack.

Infix to Postfix Conversion

ALGORITHM

6. If a Right parenthesis is encountered, then :

    a) Repeatedly pop from Stack and add to P each operator until a Left parenthesis is encountered**.**

    b) Remove the left Parenthesis.

7. Exit.

a - (*b* + *c* * *d*)/*e* to

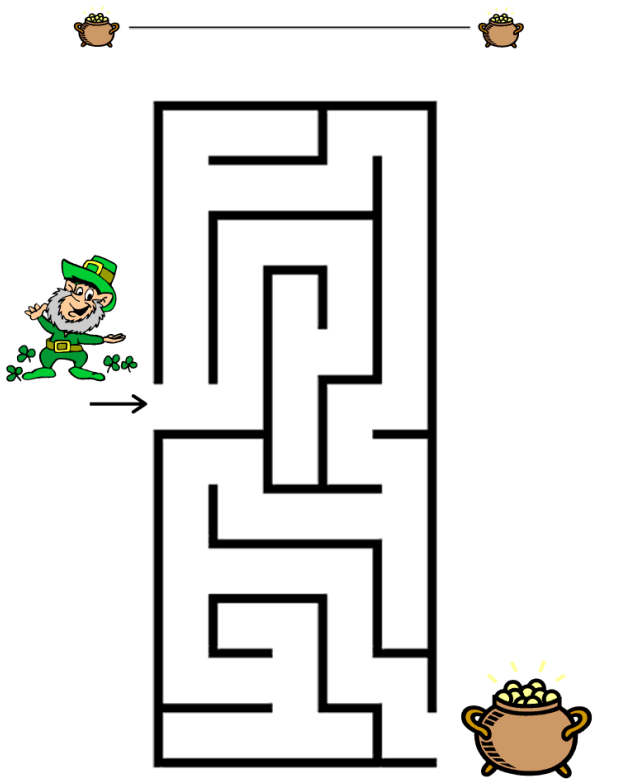
ch	stack (bottom to top)	postfixExp
a		a
−	−	a
(	−(	a
b	−(	ab
+	−(+	ab
c	−(+	abc
*	−(+*	abc
d	−(+*	abcd
)	−(+	abcd*
	−(	abcd*+
	−	abcd*+
/	−/	abcd*+
e	−/	abcd*+e
		abcd*+e/−

مثال: برای اینکه عبارت میانوندی زیر به فرم پسوندی تبدیل شود، حداقل اندازه پشته چقدر باید باشد؟

$$((a * b) - (\frac{c}{d^e})) == (f + g)$$

کاربردهای پشته

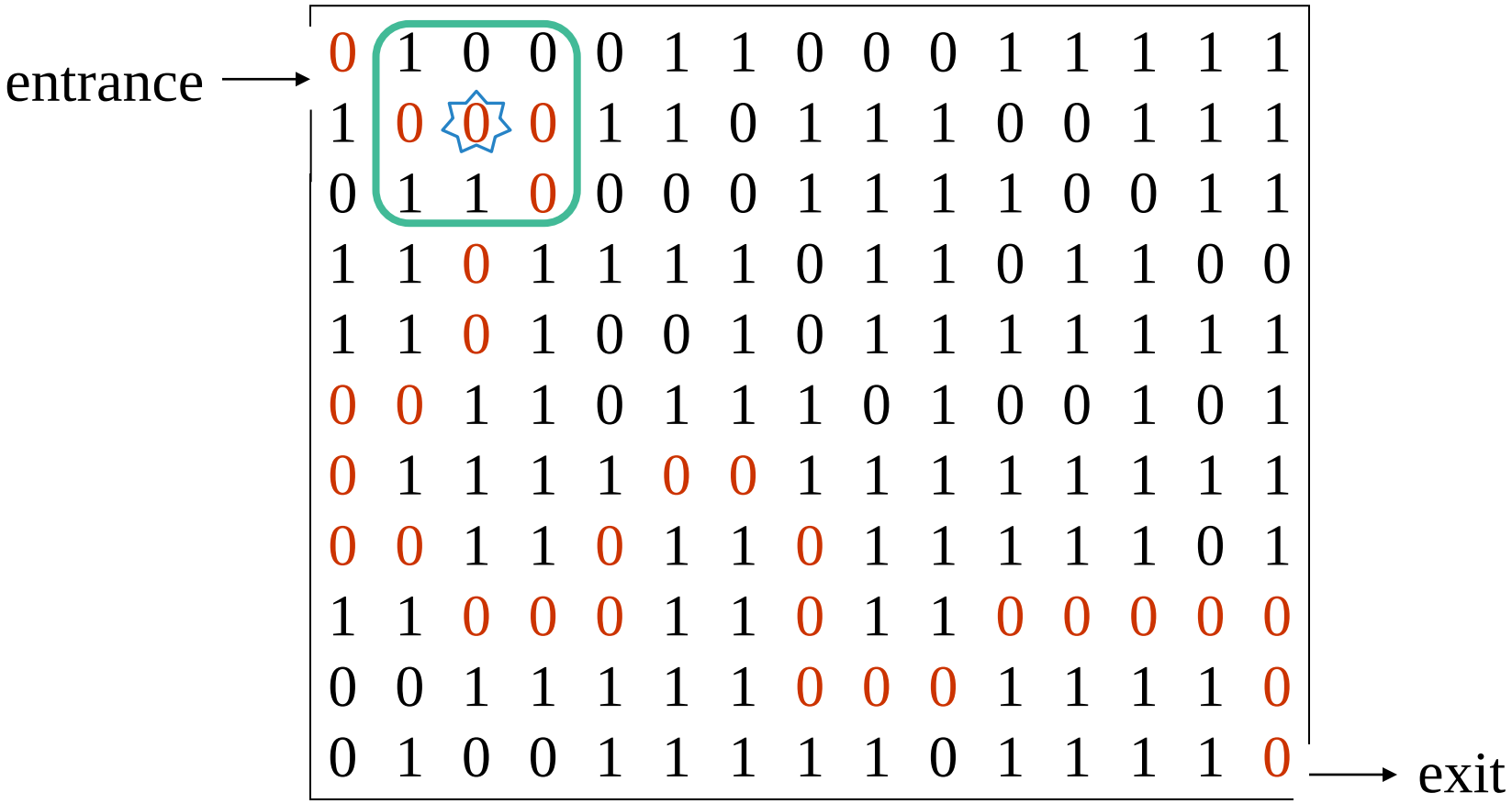
- مدیریت فراخوانی توابع
- تبدیل عبارت میانوندی به پسوندی
- مسئله Maze
- ارزشیابی عبارت های پسوندی



St. Patrick's Day Preschool Maze - Pre-Writing Skill

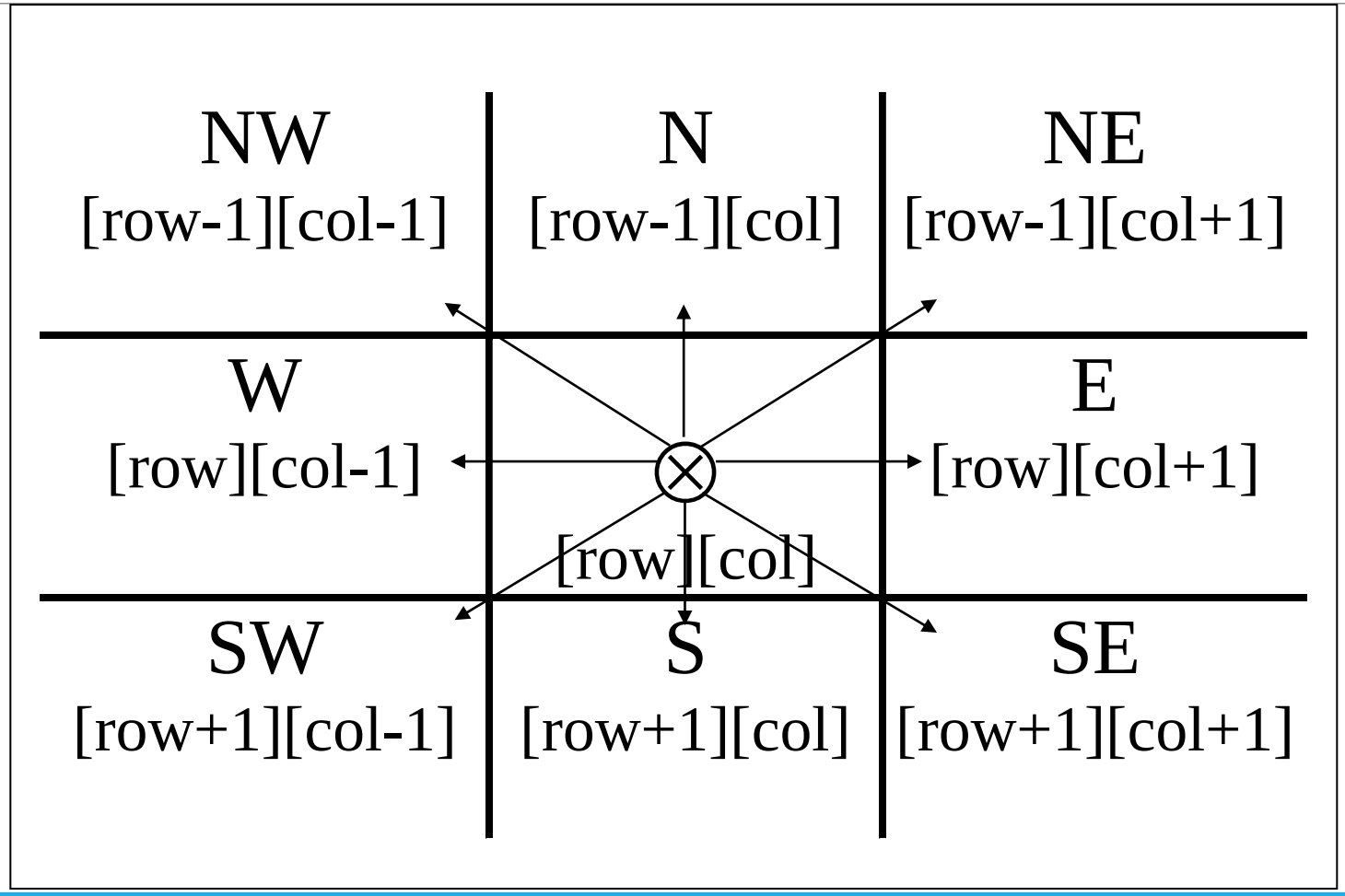
www.first-school.ws

A Mazing Problem



1: blocked path 0: through path

یک راه حل ممکن:



بکارگیری پشته برای نگهداری مسیرهای جانبی

```
struct element{
        int row;
        int col;
        int dir;
        } element;
element stack[MAX_STACK_SIZE];
```

کاربردهای پشته

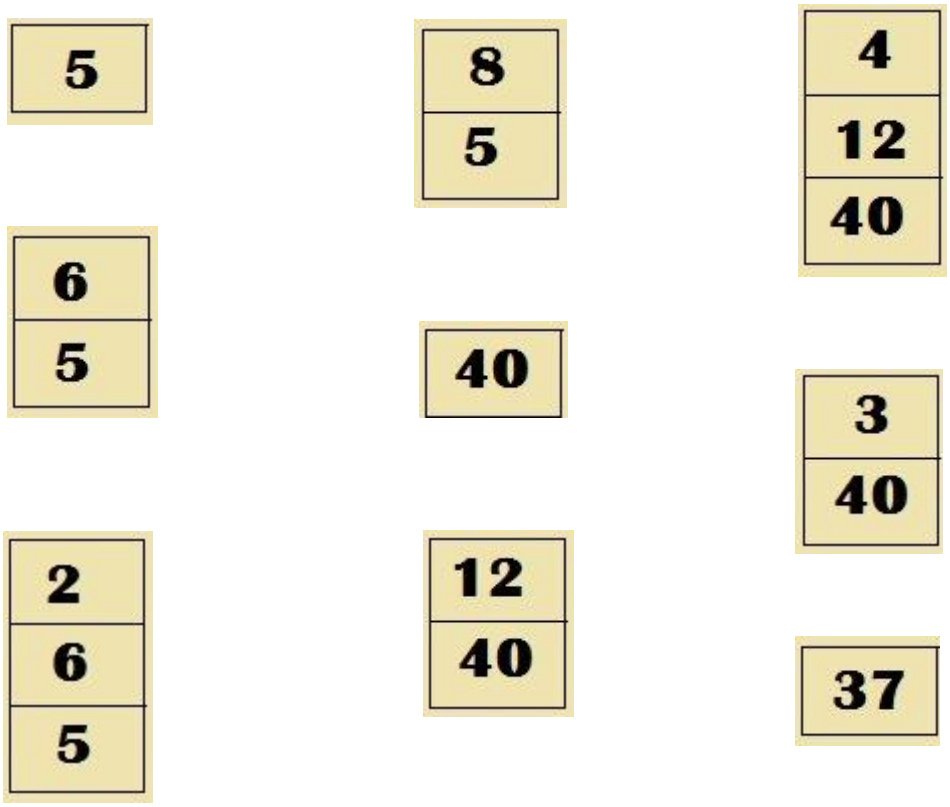
- مدیریت فراخوانی توابع
- تبدیل عبارت میانوندی به پسوندی
- مسئله Maze
- ارزشیابی عبارت های پسوندی

ارزشیابی عبارت های پسوندی

1. Add a right parenthesis “)“ at the end of P.
2. Scan p from left to right and repeat step 3 & 4 for each until the sentinel “)” is encountered.
3. If an operand is encountered, put it on Stack.
4. If an operator ¤ is encountered, then :
    a) Pop the two top elements of Stack, Say A(Top most) and B(Second Top).
    b) Evaluate A ¤ B.
    c) Push the result in Stack.
5. Set Value equal to top element of Stack.
6. Exit.

EXAMPLE

P : 5 6 2 + * 12 4 / -



As You can see now P is empty i.e. there is no more element in P. So, Value at Stack[TOP] is your answer.

In this case it is 37