STACKH.pdf

Ferre18

Programación de Sistemas y Concurrencia

2º Grado en Ingeniería Informática

Escuela Técnica Superior de Ingeniería Informática
Universidad de Málaga

```
1  //EMPIEZA AQUI
2
3
4  #ifndef STACK_H_
5  #define STACK_H_
6
7  typedef struct Node * T_Stack;
8  typedef struct Node {
9    struct Node * next;
10   int number;
11 } T_Node;
12
13 // Creates an empty stack.
14 T_Stack create();
15
16 // Returns true if the stack is empty and false in other case.
17 int isEmpty(T_Stack q);
18
19 // Inserts a number into the stack.
20 void push(T_Stack * pq, int operand);
21
22 // "Inserts" an operator into the stack and operates.
23 // Returns true if everything OK or false in other case.
24 int pushOperator(T_Stack * pq, char operator);
25
26 // Puts into data the number on top of the stack, and removes the top.
27 // Returns true if everything OK or false in other case.
28 int pop(T_Stack * pq, int * data);
29
30 // Frees the memory of a stack and sets it to empty.
31 void destroy(T_Stack * pq);
32
33 #endif /* STACK_H_ */
34
35
36 //TERMINA AQUI
```

```
1  //EMPIEZA AQUI
2
3
4  #ifndef STACK_H_
5  #define STACK_H_
6
7  typedef struct Node * T_Stack;
8  typedef struct Node {
9    struct Node * next;
10   int number;
11 } T_Node;
12
13 // Creates an empty stack.
14 T_Stack create();
15
16 // Returns true if the stack is empty and false in other case.
17 int isEmpty(T_Stack q);
18
19 // Inserts a number into the stack.
20 void push(T_Stack * pq, int operand);
21
22 // "Inserts" an operator into the stack and operates.
23 // Returns true if everything OK or false in other case.
24 int pushOperator(T_Stack * pq, char operator);
25
26 // Puts into data the number on top of the stack, and removes the top.
27 // Returns true if everything OK or false in other case.
28 int pop(T_Stack * pq, int * data);
29
30 // Frees the memory of a stack and sets it to empty.
31 void destroy(T_Stack * pq);
32
33 #endif /* STACK_H_ */
34
35
36 //TERMINA AQUI
```

```
1  //EMPIEZA AQUI
2
3
4  #ifndef STACK_H_
5  #define STACK_H_
6
7  typedef struct Node * T_Stack;
8  typedef struct Node {
9    struct Node * next;
10   int number;
11 } T_Node;
12
13 // Creates an empty stack.
14 T_Stack create();
15
16 // Returns true if the stack is empty and false in other case.
17 int isEmpty(T_Stack q);
18
19 // Inserts a number into the stack.
20 void push(T_Stack * pq, int operand);
21
22 // "Inserts" an operator into the stack and operates.
23 // Returns true if everything OK or false in other case.
24 int pushOperator(T_Stack * pq, char operator);
25
26 // Puts into data the number on top of the stack, and removes the top.
27 // Returns true if everything OK or false in other case.
28 int pop(T_Stack * pq, int * data);
29
30 // Frees the memory of a stack and sets it to empty.
31 void destroy(T_Stack * pq);
32
33 #endif /* STACK_H_ */
34
35
36 //TERMINA AQUI
```

```
1  //EMPIEZA AQUI
2
3
4  #ifndef STACK_H_
5  #define STACK_H_
6
7  typedef struct Node * T_Stack;
8  typedef struct Node {
9    struct Node * next;
10   int number;
11 } T_Node;
12
13 // Creates an empty stack.
14 T_Stack create();
15
16 // Returns true if the stack is empty and false in other case.
17 int isEmpty(T_Stack q);
18
19 // Inserts a number into the stack.
20 void push(T_Stack * pq, int operand);
21
22 // "Inserts" an operator into the stack and operates.
23 // Returns true if everything OK or false in other case.
24 int pushOperator(T_Stack * pq, char operator);
25
26 // Puts into data the number on top of the stack, and removes the top.
27 // Returns true if everything OK or false in other case.
28 int pop(T_Stack * pq, int * data);
29
30 // Frees the memory of a stack and sets it to empty.
31 void destroy(T_Stack * pq);
32
33 #endif /* STACK_H_ */
34
35
36 //TERMINA AQUI
```

```
1  //EMPIEZA AQUI
2
3
4  #ifndef STACK_H_
5  #define STACK_H_
6
7  typedef struct Node * T_Stack;
8  typedef struct Node {
9    struct Node * next;
10   int number;
11 } T_Node;
12
13 // Creates an empty stack.
14 T_Stack create();
15
16 // Returns true if the stack is empty and false in other case.
17 int isEmpty(T_Stack q);
18
19 // Inserts a number into the stack.
20 void push(T_Stack * pq, int operand);
21
22 // "Inserts" an operator into the stack and operates.
23 // Returns true if everything OK or false in other case.
24 int pushOperator(T_Stack * pq, char operator);
25
26 // Puts into data the number on top of the stack, and removes the top.
27 // Returns true if everything OK or false in other case.
28 int pop(T_Stack * pq, int * data);
29
30 // Frees the memory of a stack and sets it to empty.
31 void destroy(T_Stack * pq);
32
33 #endif /* STACK_H_ */
34
35
36 //TERMINA AQUI
```

```
1  //EMPIEZA AQUI
2
3
4  #ifndef STACK_H_
5  #define STACK_H_
6
7  typedef struct Node * T_Stack;
8  typedef struct Node {
9    struct Node * next;
10   int number;
11 } T_Node;
12
13 // Creates an empty stack.
14 T_Stack create();
15
16 // Returns true if the stack is empty and false in other case.
17 int isEmpty(T_Stack q);
18
19 // Inserts a number into the stack.
20 void push(T_Stack * pq, int operand);
21
22 // "Inserts" an operator into the stack and operates.
23 // Returns true if everything OK or false in other case.
24 int pushOperator(T_Stack * pq, char operator);
25
26 // Puts into data the number on top of the stack, and removes the top.
27 // Returns true if everything OK or false in other case.
28 int pop(T_Stack * pq, int * data);
29
30 // Frees the memory of a stack and sets it to empty.
31 void destroy(T_Stack * pq);
32
33 #endif /* STACK_H_ */
34
35
36 //TERMINA AQUI
```

```
1  //EMPIEZA AQUI
2
3
4  #ifndef STACK_H_
5  #define STACK_H_
6
7  typedef struct Node * T_Stack;
8  typedef struct Node {
9    struct Node * next;
10   int number;
11 } T_Node;
12
13 // Creates an empty stack.
14 T_Stack create();
15
16 // Returns true if the stack is empty and false in other case.
17 int isEmpty(T_Stack q);
18
19 // Inserts a number into the stack.
20 void push(T_Stack * pq, int operand);
21
22 // "Inserts" an operator into the stack and operates.
23 // Returns true if everything OK or false in other case.
24 int pushOperator(T_Stack * pq, char operator);
25
26 // Puts into data the number on top of the stack, and removes the top.
27 // Returns true if everything OK or false in other case.
28 int pop(T_Stack * pq, int * data);
29
30 // Frees the memory of a stack and sets it to empty.
31 void destroy(T_Stack * pq);
32
33 #endif /* STACK_H_ */
34
35
36 //TERMINA AQUI
```

```
1  //EMPIEZA AQUI
2
3
4  #ifndef STACK_H_
5  #define STACK_H_
6
7  typedef struct Node * T_Stack;
8  typedef struct Node {
9    struct Node * next;
10   int number;
11 } T_Node;
12
13 // Creates an empty stack.
14 T_Stack create();
15
16 // Returns true if the stack is empty and false in other case.
17 int isEmpty(T_Stack q);
18
19 // Inserts a number into the stack.
20 void push(T_Stack * pq, int operand);
21
22 // "Inserts" an operator into the stack and operates.
23 // Returns true if everything OK or false in other case.
24 int pushOperator(T_Stack * pq, char operator);
25
26 // Puts into data the number on top of the stack, and removes the top.
27 // Returns true if everything OK or false in other case.
28 int pop(T_Stack * pq, int * data);
29
30 // Frees the memory of a stack and sets it to empty.
31 void destroy(T_Stack * pq);
32
33 #endif /* STACK_H_ */
34
35
36 //TERMINA AQUI
```

```
1  //EMPIEZA AQUI
2
3
4  #ifndef STACK_H_
5  #define STACK_H_
6
7  typedef struct Node * T_Stack;
8  typedef struct Node {
9    struct Node * next;
10   int number;
11 } T_Node;
12
13 // Creates an empty stack.
14 T_Stack create();
15
16 // Returns true if the stack is empty and false in other case.
17 int isEmpty(T_Stack q);
18
19 // Inserts a number into the stack.
20 void push(T_Stack * pq, int operand);
21
22 // "Inserts" an operator into the stack and operates.
23 // Returns true if everything OK or false in other case.
24 int pushOperator(T_Stack * pq, char operator);
25
26 // Puts into data the number on top of the stack, and removes the top.
27 // Returns true if everything OK or false in other case.
28 int pop(T_Stack * pq, int * data);
29
30 // Frees the memory of a stack and sets it to empty.
31 void destroy(T_Stack * pq);
32
33 #endif /* STACK_H_ */
34
35
36 //TERMINA AQUI
```

```
1  //EMPIEZA AQUI
2
3
4  #ifndef STACK_H_
5  #define STACK_H_
6
7  typedef struct Node * T_Stack;
8  typedef struct Node {
9    struct Node * next;
10   int number;
11 } T_Node;
12
13 // Creates an empty stack.
14 T_Stack create();
15
16 // Returns true if the stack is empty and false in other case.
17 int isEmpty(T_Stack q);
18
19 // Inserts a number into the stack.
20 void push(T_Stack * pq, int operand);
21
22 // "Inserts" an operator into the stack and operates.
23 // Returns true if everything OK or false in other case.
24 int pushOperator(T_Stack * pq, char operator);
25
26 // Puts into data the number on top of the stack, and removes the top.
27 // Returns true if everything OK or false in other case.
28 int pop(T_Stack * pq, int * data);
29
30 // Frees the memory of a stack and sets it to empty.
31 void destroy(T_Stack * pq);
32
33 #endif /* STACK_H_ */
34
35
36 //TERMINA AQUI
```

```
1  //EMPIEZA AQUI
2
3
4  #ifndef STACK_H_
5  #define STACK_H_
6
7  typedef struct Node * T_Stack;
8  typedef struct Node {
9    struct Node * next;
10   int number;
11 } T_Node;
12
13 // Creates an empty stack.
14 T_Stack create();
15
16 // Returns true if the stack is empty and false in other case.
17 int isEmpty(T_Stack q);
18
19 // Inserts a number into the stack.
20 void push(T_Stack * pq, int operand);
21
22 // "Inserts" an operator into the stack and operates.
23 // Returns true if everything OK or false in other case.
24 int pushOperator(T_Stack * pq, char operator);
25
26 // Puts into data the number on top of the stack, and removes the top.
27 // Returns true if everything OK or false in other case.
28 int pop(T_Stack * pq, int * data);
29
30 // Frees the memory of a stack and sets it to empty.
31 void destroy(T_Stack * pq);
32
33 #endif /* STACK_H_ */
34
35
36 //TERMINA AQUI
```

```
1  //EMPIEZA AQUI
2
3
4  #ifndef STACK_H_
5  #define STACK_H_
6
7  typedef struct Node * T_Stack;
8  typedef struct Node {
9    struct Node * next;
10   int number;
11 } T_Node;
12
13 // Creates an empty stack.
14 T_Stack create();
15
16 // Returns true if the stack is empty and false in other case.
17 int isEmpty(T_Stack q);
18
19 // Inserts a number into the stack.
20 void push(T_Stack * pq, int operand);
21
22 // "Inserts" an operator into the stack and operates.
23 // Returns true if everything OK or false in other case.
24 int pushOperator(T_Stack * pq, char operator);
25
26 // Puts into data the number on top of the stack, and removes the top.
27 // Returns true if everything OK or false in other case.
28 int pop(T_Stack * pq, int * data);
29
30 // Frees the memory of a stack and sets it to empty.
31 void destroy(T_Stack * pq);
32
33 #endif /* STACK_H_ */
34
35
36 //TERMINA AQUI
```

```
1  //EMPIEZA AQUI
2
3
4  #ifndef STACK_H_
5  #define STACK_H_
6
7  typedef struct Node * T_Stack;
8  typedef struct Node {
9    struct Node * next;
10   int number;
11 } T_Node;
12
13 // Creates an empty stack.
14 T_Stack create();
15
16 // Returns true if the stack is empty and false in other case.
17 int isEmpty(T_Stack q);
18
19 // Inserts a number into the stack.
20 void push(T_Stack * pq, int operand);
21
22 // "Inserts" an operator into the stack and operates.
23 // Returns true if everything OK or false in other case.
24 int pushOperator(T_Stack * pq, char operator);
25
26 // Puts into data the number on top of the stack, and removes the top.
27 // Returns true if everything OK or false in other case.
28 int pop(T_Stack * pq, int * data);
29
30 // Frees the memory of a stack and sets it to empty.
31 void destroy(T_Stack * pq);
32
33 #endif /* STACK_H_ */
34
35
36 //TERMINA AQUI
```

```
1  //EMPIEZA AQUI
2
3
4  #ifndef STACK_H_
5  #define STACK_H_
6
7  typedef struct Node * T_Stack;
8  typedef struct Node {
9    struct Node * next;
10   int number;
11 } T_Node;
12
13 // Creates an empty stack.
14 T_Stack create();
15
16 // Returns true if the stack is empty and false in other case.
17 int isEmpty(T_Stack q);
18
19 // Inserts a number into the stack.
20 void push(T_Stack * pq, int operand);
21
22 // "Inserts" an operator into the stack and operates.
23 // Returns true if everything OK or false in other case.
24 int pushOperator(T_Stack * pq, char operator);
25
26 // Puts into data the number on top of the stack, and removes the top.
27 // Returns true if everything OK or false in other case.
28 int pop(T_Stack * pq, int * data);
29
30 // Frees the memory of a stack and sets it to empty.
31 void destroy(T_Stack * pq);
32
33 #endif /* STACK_H_ */
34
35
36 //TERMINA AQUI
```

```
1  //EMPIEZA AQUI
2
3
4  #ifndef STACK_H_
5  #define STACK_H_
6
7  typedef struct Node * T_Stack;
8  typedef struct Node {
9    struct Node * next;
10   int number;
11 } T_Node;
12
13 // Creates an empty stack.
14 T_Stack create();
15
16 // Returns true if the stack is empty and false in other case.
17 int isEmpty(T_Stack q);
18
19 // Inserts a number into the stack.
20 void push(T_Stack * pq, int operand);
21
22 // "Inserts" an operator into the stack and operates.
23 // Returns true if everything OK or false in other case.
24 int pushOperator(T_Stack * pq, char operator);
25
26 // Puts into data the number on top of the stack, and removes the top.
27 // Returns true if everything OK or false in other case.
28 int pop(T_Stack * pq, int * data);
29
30 // Frees the memory of a stack and sets it to empty.
31 void destroy(T_Stack * pq);
32
33 #endif /* STACK_H_ */
34
35
36 //TERMINA AQUI
```

```
1  //EMPIEZA AQUI
2
3
4  #ifndef STACK_H_
5  #define STACK_H_
6
7  typedef struct Node * T_Stack;
8  typedef struct Node {
9    struct Node * next;
10   int number;
11 } T_Node;
12
13 // Creates an empty stack.
14 T_Stack create();
15
16 // Returns true if the stack is empty and false in other case.
17 int isEmpty(T_Stack q);
18
19 // Inserts a number into the stack.
20 void push(T_Stack * pq, int operand);
21
22 // "Inserts" an operator into the stack and operates.
23 // Returns true if everything OK or false in other case.
24 int pushOperator(T_Stack * pq, char operator);
25
26 // Puts into data the number on top of the stack, and removes the top.
27 // Returns true if everything OK or false in other case.
28 int pop(T_Stack * pq, int * data);
29
30 // Frees the memory of a stack and sets it to empty.
31 void destroy(T_Stack * pq);
32
33 #endif /* STACK_H_ */
34
35
36 //TERMINA AQUI
```

```
1  //EMPIEZA AQUI
2
3
4  #ifndef STACK_H_
5  #define STACK_H_
6
7  typedef struct Node * T_Stack;
8  typedef struct Node {
9    struct Node * next;
10   int number;
11 } T_Node;
12
13 // Creates an empty stack.
14 T_Stack create();
15
16 // Returns true if the stack is empty and false in other case.
17 int isEmpty(T_Stack q);
18
19 // Inserts a number into the stack.
20 void push(T_Stack * pq, int operand);
21
22 // "Inserts" an operator into the stack and operates.
23 // Returns true if everything OK or false in other case.
24 int pushOperator(T_Stack * pq, char operator);
25
26 // Puts into data the number on top of the stack, and removes the top.
27 // Returns true if everything OK or false in other case.
28 int pop(T_Stack * pq, int * data);
29
30 // Frees the memory of a stack and sets it to empty.
31 void destroy(T_Stack * pq);
32
33 #endif /* STACK_H_ */
34
35
36 //TERMINA AQUI
```

```
1  //EMPIEZA AQUI
2
3
4  #ifndef STACK_H_
5  #define STACK_H_
6
7  typedef struct Node * T_Stack;
8  typedef struct Node {
9    struct Node * next;
10   int number;
11 } T_Node;
12
13 // Creates an empty stack.
14 T_Stack create();
15
16 // Returns true if the stack is empty and false in other case.
17 int isEmpty(T_Stack q);
18
19 // Inserts a number into the stack.
20 void push(T_Stack * pq, int operand);
21
22 // "Inserts" an operator into the stack and operates.
23 // Returns true if everything OK or false in other case.
24 int pushOperator(T_Stack * pq, char operator);
25
26 // Puts into data the number on top of the stack, and removes the top.
27 // Returns true if everything OK or false in other case.
28 int pop(T_Stack * pq, int * data);
29
30 // Frees the memory of a stack and sets it to empty.
31 void destroy(T_Stack * pq);
32
33 #endif /* STACK_H_ */
34
35
36 //TERMINA AQUI
```

```
1  //EMPIEZA AQUI
2
3
4  #ifndef STACK_H_
5  #define STACK_H_
6
7  typedef struct Node * T_Stack;
8  typedef struct Node {
9    struct Node * next;
10   int number;
11 } T_Node;
12
13 // Creates an empty stack.
14 T_Stack create();
15
16 // Returns true if the stack is empty and false in other case.
17 int isEmpty(T_Stack q);
18
19 // Inserts a number into the stack.
20 void push(T_Stack * pq, int operand);
21
22 // "Inserts" an operator into the stack and operates.
23 // Returns true if everything OK or false in other case.
24 int pushOperator(T_Stack * pq, char operator);
25
26 // Puts into data the number on top of the stack, and removes the top.
27 // Returns true if everything OK or false in other case.
28 int pop(T_Stack * pq, int * data);
29
30 // Frees the memory of a stack and sets it to empty.
31 void destroy(T_Stack * pq);
32
33 #endif /* STACK_H_ */
34
35
36 //TERMINA AQUI
```

```
1  //EMPIEZA AQUI
2
3
4  #ifndef STACK_H_
5  #define STACK_H_
6
7  typedef struct Node * T_Stack;
8  typedef struct Node {
9    struct Node * next;
10   int number;
11 } T_Node;
12
13 // Creates an empty stack.
14 T_Stack create();
15
16 // Returns true if the stack is empty and false in other case.
17 int isEmpty(T_Stack q);
18
19 // Inserts a number into the stack.
20 void push(T_Stack * pq, int operand);
21
22 // "Inserts" an operator into the stack and operates.
23 // Returns true if everything OK or false in other case.
24 int pushOperator(T_Stack * pq, char operator);
25
26 // Puts into data the number on top of the stack, and removes the top.
27 // Returns true if everything OK or false in other case.
28 int pop(T_Stack * pq, int * data);
29
30 // Frees the memory of a stack and sets it to empty.
31 void destroy(T_Stack * pq);
32
33 #endif /* STACK_H_ */
34
35
36 //TERMINA AQUI
```

```
1  //EMPIEZA AQUI
2
3
4  #ifndef STACK_H_
5  #define STACK_H_
6
7  typedef struct Node * T_Stack;
8  typedef struct Node {
9    struct Node * next;
10   int number;
11 } T_Node;
12
13 // Creates an empty stack.
14 T_Stack create();
15
16 // Returns true if the stack is empty and false in other case.
17 int isEmpty(T_Stack q);
18
19 // Inserts a number into the stack.
20 void push(T_Stack * pq, int operand);
21
22 // "Inserts" an operator into the stack and operates.
23 // Returns true if everything OK or false in other case.
24 int pushOperator(T_Stack * pq, char operator);
25
26 // Puts into data the number on top of the stack, and removes the top.
27 // Returns true if everything OK or false in other case.
28 int pop(T_Stack * pq, int * data);
29
30 // Frees the memory of a stack and sets it to empty.
31 void destroy(T_Stack * pq);
32
33 #endif /* STACK_H_ */
34
35
36 //TERMINA AQUI
```

```
1  //EMPIEZA AQUI
2
3
4  #ifndef STACK_H_
5  #define STACK_H_
6
7  typedef struct Node * T_Stack;
8  typedef struct Node {
9    struct Node * next;
10   int number;
11 } T_Node;
12
13 // Creates an empty stack.
14 T_Stack create();
15
16 // Returns true if the stack is empty and false in other case.
17 int isEmpty(T_Stack q);
18
19 // Inserts a number into the stack.
20 void push(T_Stack * pq, int operand);
21
22 // "Inserts" an operator into the stack and operates.
23 // Returns true if everything OK or false in other case.
24 int pushOperator(T_Stack * pq, char operator);
25
26 // Puts into data the number on top of the stack, and removes the top.
27 // Returns true if everything OK or false in other case.
28 int pop(T_Stack * pq, int * data);
29
30 // Frees the memory of a stack and sets it to empty.
31 void destroy(T_Stack * pq);
32
33 #endif /* STACK_H_ */
34
35
36 //TERMINA AQUI
```

```
1  //EMPIEZA AQUI
2
3
4  #ifndef STACK_H_
5  #define STACK_H_
6
7  typedef struct Node * T_Stack;
8  typedef struct Node {
9    struct Node * next;
10   int number;
11 } T_Node;
12
13 // Creates an empty stack.
14 T_Stack create();
15
16 // Returns true if the stack is empty and false in other case.
17 int isEmpty(T_Stack q);
18
19 // Inserts a number into the stack.
20 void push(T_Stack * pq, int operand);
21
22 // "Inserts" an operator into the stack and operates.
23 // Returns true if everything OK or false in other case.
24 int pushOperator(T_Stack * pq, char operator);
25
26 // Puts into data the number on top of the stack, and removes the top.
27 // Returns true if everything OK or false in other case.
28 int pop(T_Stack * pq, int * data);
29
30 // Frees the memory of a stack and sets it to empty.
31 void destroy(T_Stack * pq);
32
33 #endif /* STACK_H_ */
34
35
36 //TERMINA AQUI
```

```
1  //EMPIEZA AQUI
2
3
4  #ifndef STACK_H_
5  #define STACK_H_
6
7  typedef struct Node * T_Stack;
8  typedef struct Node {
9    struct Node * next;
10   int number;
11 } T_Node;
12
13 // Creates an empty stack.
14 T_Stack create();
15
16 // Returns true if the stack is empty and false in other case.
17 int isEmpty(T_Stack q);
18
19 // Inserts a number into the stack.
20 void push(T_Stack * pq, int operand);
21
22 // "Inserts" an operator into the stack and operates.
23 // Returns true if everything OK or false in other case.
24 int pushOperator(T_Stack * pq, char operator);
25
26 // Puts into data the number on top of the stack, and removes the top.
27 // Returns true if everything OK or false in other case.
28 int pop(T_Stack * pq, int * data);
29
30 // Frees the memory of a stack and sets it to empty.
31 void destroy(T_Stack * pq);
32
33 #endif /* STACK_H_ */
34
35
36 //TERMINA AQUI
```

```
1  //EMPIEZA AQUI
2
3
4  #ifndef STACK_H_
5  #define STACK_H_
6
7  typedef struct Node * T_Stack;
8  typedef struct Node {
9    struct Node * next;
10   int number;
11 } T_Node;
12
13 // Creates an empty stack.
14 T_Stack create();
15
16 // Returns true if the stack is empty and false in other case.
17 int isEmpty(T_Stack q);
18
19 // Inserts a number into the stack.
20 void push(T_Stack * pq, int operand);
21
22 // "Inserts" an operator into the stack and operates.
23 // Returns true if everything OK or false in other case.
24 int pushOperator(T_Stack * pq, char operator);
25
26 // Puts into data the number on top of the stack, and removes the top.
27 // Returns true if everything OK or false in other case.
28 int pop(T_Stack * pq, int * data);
29
30 // Frees the memory of a stack and sets it to empty.
31 void destroy(T_Stack * pq);
32
33 #endif /* STACK_H_ */
34
35
36 //TERMINA AQUI
```

```
1  //EMPIEZA AQUI
2
3
4  #ifndef STACK_H_
5  #define STACK_H_
6
7  typedef struct Node * T_Stack;
8  typedef struct Node {
9    struct Node * next;
10   int number;
11 } T_Node;
12
13 // Creates an empty stack.
14 T_Stack create();
15
16 // Returns true if the stack is empty and false in other case.
17 int isEmpty(T_Stack q);
18
19 // Inserts a number into the stack.
20 void push(T_Stack * pq, int operand);
21
22 // "Inserts" an operator into the stack and operates.
23 // Returns true if everything OK or false in other case.
24 int pushOperator(T_Stack * pq, char operator);
25
26 // Puts into data the number on top of the stack, and removes the top.
27 // Returns true if everything OK or false in other case.
28 int pop(T_Stack * pq, int * data);
29
30 // Frees the memory of a stack and sets it to empty.
31 void destroy(T_Stack * pq);
32
33 #endif /* STACK_H_ */
34
35
36 //TERMINA AQUI
```

```
1  //EMPIEZA AQUI
2
3
4  #ifndef STACK_H_
5  #define STACK_H_
6
7  typedef struct Node * T_Stack;
8  typedef struct Node {
9    struct Node * next;
10   int number;
11 } T_Node;
12
13 // Creates an empty stack.
14 T_Stack create();
15
16 // Returns true if the stack is empty and false in other case.
17 int isEmpty(T_Stack q);
18
19 // Inserts a number into the stack.
20 void push(T_Stack * pq, int operand);
21
22 // "Inserts" an operator into the stack and operates.
23 // Returns true if everything OK or false in other case.
24 int pushOperator(T_Stack * pq, char operator);
25
26 // Puts into data the number on top of the stack, and removes the top.
27 // Returns true if everything OK or false in other case.
28 int pop(T_Stack * pq, int * data);
29
30 // Frees the memory of a stack and sets it to empty.
31 void destroy(T_Stack * pq);
32
33 #endif /* STACK_H_ */
34
35
36 //TERMINA AQUI
```

```
1  //EMPIEZA AQUI
2
3
4  #ifndef STACK_H_
5  #define STACK_H_
6
7  typedef struct Node * T_Stack;
8  typedef struct Node {
9    struct Node * next;
10   int number;
11 } T_Node;
12
13 // Creates an empty stack.
14 T_Stack create();
15
16 // Returns true if the stack is empty and false in other case.
17 int isEmpty(T_Stack q);
18
19 // Inserts a number into the stack.
20 void push(T_Stack * pq, int operand);
21
22 // "Inserts" an operator into the stack and operates.
23 // Returns true if everything OK or false in other case.
24 int pushOperator(T_Stack * pq, char operator);
25
26 // Puts into data the number on top of the stack, and removes the top.
27 // Returns true if everything OK or false in other case.
28 int pop(T_Stack * pq, int * data);
29
30 // Frees the memory of a stack and sets it to empty.
31 void destroy(T_Stack * pq);
32
33 #endif /* STACK_H_ */
34
35
36 //TERMINA AQUI
```

```
1  //EMPIEZA AQUI
2
3
4  #ifndef STACK_H_
5  #define STACK_H_
6
7  typedef struct Node * T_Stack;
8  typedef struct Node {
9    struct Node * next;
10   int number;
11 } T_Node;
12
13 // Creates an empty stack.
14 T_Stack create();
15
16 // Returns true if the stack is empty and false in other case.
17 int isEmpty(T_Stack q);
18
19 // Inserts a number into the stack.
20 void push(T_Stack * pq, int operand);
21
22 // "Inserts" an operator into the stack and operates.
23 // Returns true if everything OK or false in other case.
24 int pushOperator(T_Stack * pq, char operator);
25
26 // Puts into data the number on top of the stack, and removes the top.
27 // Returns true if everything OK or false in other case.
28 int pop(T_Stack * pq, int * data);
29
30 // Frees the memory of a stack and sets it to empty.
31 void destroy(T_Stack * pq);
32
33 #endif /* STACK_H_ */
34
35
36 //TERMINA AQUI
```

```
1  //EMPIEZA AQUI
2
3
4  #ifndef STACK_H_
5  #define STACK_H_
6
7  typedef struct Node * T_Stack;
8  typedef struct Node {
9    struct Node * next;
10   int number;
11 } T_Node;
12
13 // Creates an empty stack.
14 T_Stack create();
15
16 // Returns true if the stack is empty and false in other case.
17 int isEmpty(T_Stack q);
18
19 // Inserts a number into the stack.
20 void push(T_Stack * pq, int operand);
21
22 // "Inserts" an operator into the stack and operates.
23 // Returns true if everything OK or false in other case.
24 int pushOperator(T_Stack * pq, char operator);
25
26 // Puts into data the number on top of the stack, and removes the top.
27 // Returns true if everything OK or false in other case.
28 int pop(T_Stack * pq, int * data);
29
30 // Frees the memory of a stack and sets it to empty.
31 void destroy(T_Stack * pq);
32
33 #endif /* STACK_H_ */
34
35
36 //TERMINA AQUI
```

```
1  //EMPIEZA AQUI
2
3
4  #ifndef STACK_H_
5  #define STACK_H_
6
7  typedef struct Node * T_Stack;
8  typedef struct Node {
9    struct Node * next;
10   int number;
11 } T_Node;
12
13 // Creates an empty stack.
14 T_Stack create();
15
16 // Returns true if the stack is empty and false in other case.
17 int isEmpty(T_Stack q);
18
19 // Inserts a number into the stack.
20 void push(T_Stack * pq, int operand);
21
22 // "Inserts" an operator into the stack and operates.
23 // Returns true if everything OK or false in other case.
24 int pushOperator(T_Stack * pq, char operator);
25
26 // Puts into data the number on top of the stack, and removes the top.
27 // Returns true if everything OK or false in other case.
28 int pop(T_Stack * pq, int * data);
29
30 // Frees the memory of a stack and sets it to empty.
31 void destroy(T_Stack * pq);
32
33 #endif /* STACK_H_ */
34
35
36 //TERMINA AQUI
```

```
1  //EMPIEZA AQUI
2
3
4  #ifndef STACK_H_
5  #define STACK_H_
6
7  typedef struct Node * T_Stack;
8  typedef struct Node {
9    struct Node * next;
10   int number;
11 } T_Node;
12
13 // Creates an empty stack.
14 T_Stack create();
15
16 // Returns true if the stack is empty and false in other case.
17 int isEmpty(T_Stack q);
18
19 // Inserts a number into the stack.
20 void push(T_Stack * pq, int operand);
21
22 // "Inserts" an operator into the stack and operates.
23 // Returns true if everything OK or false in other case.
24 int pushOperator(T_Stack * pq, char operator);
25
26 // Puts into data the number on top of the stack, and removes the top.
27 // Returns true if everything OK or false in other case.
28 int pop(T_Stack * pq, int * data);
29
30 // Frees the memory of a stack and sets it to empty.
31 void destroy(T_Stack * pq);
32
33 #endif /* STACK_H_ */
34
35
36 //TERMINA AQUI
```

```
1  //EMPIEZA AQUI
2
3
4  #ifndef STACK_H_
5  #define STACK_H_
6
7  typedef struct Node * T_Stack;
8  typedef struct Node {
9    struct Node * next;
10   int number;
11 } T_Node;
12
13 // Creates an empty stack.
14 T_Stack create();
15
16 // Returns true if the stack is empty and false in other case.
17 int isEmpty(T_Stack q);
18
19 // Inserts a number into the stack.
20 void push(T_Stack * pq, int operand);
21
22 // "Inserts" an operator into the stack and operates.
23 // Returns true if everything OK or false in other case.
24 int pushOperator(T_Stack * pq, char operator);
25
26 // Puts into data the number on top of the stack, and removes the top.
27 // Returns true if everything OK or false in other case.
28 int pop(T_Stack * pq, int * data);
29
30 // Frees the memory of a stack and sets it to empty.
31 void destroy(T_Stack * pq);
32
33 #endif /* STACK_H_ */
34
35
36 //TERMINA AQUI
```

```
1  //EMPIEZA AQUI
2
3
4  #ifndef STACK_H_
5  #define STACK_H_
6
7  typedef struct Node * T_Stack;
8  typedef struct Node {
9    struct Node * next;
10   int number;
11 } T_Node;
12
13 // Creates an empty stack.
14 T_Stack create();
15
16 // Returns true if the stack is empty and false in other case.
17 int isEmpty(T_Stack q);
18
19 // Inserts a number into the stack.
20 void push(T_Stack * pq, int operand);
21
22 // "Inserts" an operator into the stack and operates.
23 // Returns true if everything OK or false in other case.
24 int pushOperator(T_Stack * pq, char operator);
25
26 // Puts into data the number on top of the stack, and removes the top.
27 // Returns true if everything OK or false in other case.
28 int pop(T_Stack * pq, int * data);
29
30 // Frees the memory of a stack and sets it to empty.
31 void destroy(T_Stack * pq);
32
33 #endif /* STACK_H_ */
34
35
36 //TERMINA AQUI
```

```
1  //EMPIEZA AQUI
2
3
4  #ifndef STACK_H_
5  #define STACK_H_
6
7  typedef struct Node * T_Stack;
8  typedef struct Node {
9    struct Node * next;
10   int number;
11 } T_Node;
12
13 // Creates an empty stack.
14 T_Stack create();
15
16 // Returns true if the stack is empty and false in other case.
17 int isEmpty(T_Stack q);
18
19 // Inserts a number into the stack.
20 void push(T_Stack * pq, int operand);
21
22 // "Inserts" an operator into the stack and operates.
23 // Returns true if everything OK or false in other case.
24 int pushOperator(T_Stack * pq, char operator);
25
26 // Puts into data the number on top of the stack, and removes the top.
27 // Returns true if everything OK or false in other case.
28 int pop(T_Stack * pq, int * data);
29
30 // Frees the memory of a stack and sets it to empty.
31 void destroy(T_Stack * pq);
32
33 #endif /* STACK_H_ */
34
35
36 //TERMINA AQUI
```

```
1  //EMPIEZA AQUI
2
3
4  #ifndef STACK_H_
5  #define STACK_H_
6
7  typedef struct Node * T_Stack;
8  typedef struct Node {
9    struct Node * next;
10   int number;
11 } T_Node;
12
13 // Creates an empty stack.
14 T_Stack create();
15
16 // Returns true if the stack is empty and false in other case.
17 int isEmpty(T_Stack q);
18
19 // Inserts a number into the stack.
20 void push(T_Stack * pq, int operand);
21
22 // "Inserts" an operator into the stack and operates.
23 // Returns true if everything OK or false in other case.
24 int pushOperator(T_Stack * pq, char operator);
25
26 // Puts into data the number on top of the stack, and removes the top.
27 // Returns true if everything OK or false in other case.
28 int pop(T_Stack * pq, int * data);
29
30 // Frees the memory of a stack and sets it to empty.
31 void destroy(T_Stack * pq);
32
33 #endif /* STACK_H_ */
34
35
36 //TERMINA AQUI
```

```
1  //EMPIEZA AQUI
2
3
4  #ifndef STACK_H_
5  #define STACK_H_
6
7  typedef struct Node * T_Stack;
8  typedef struct Node {
9    struct Node * next;
10   int number;
11 } T_Node;
12
13 // Creates an empty stack.
14 T_Stack create();
15
16 // Returns true if the stack is empty and false in other case.
17 int isEmpty(T_Stack q);
18
19 // Inserts a number into the stack.
20 void push(T_Stack * pq, int operand);
21
22 // "Inserts" an operator into the stack and operates.
23 // Returns true if everything OK or false in other case.
24 int pushOperator(T_Stack * pq, char operator);
25
26 // Puts into data the number on top of the stack, and removes the top.
27 // Returns true if everything OK or false in other case.
28 int pop(T_Stack * pq, int * data);
29
30 // Frees the memory of a stack and sets it to empty.
31 void destroy(T_Stack * pq);
32
33 #endif /* STACK_H_ */
34
35
36 //TERMINA AQUI
```

```
1  //EMPIEZA AQUI
2
3
4  #ifndef STACK_H_
5  #define STACK_H_
6
7  typedef struct Node * T_Stack;
8  typedef struct Node {
9    struct Node * next;
10   int number;
11 } T_Node;
12
13 // Creates an empty stack.
14 T_Stack create();
15
16 // Returns true if the stack is empty and false in other case.
17 int isEmpty(T_Stack q);
18
19 // Inserts a number into the stack.
20 void push(T_Stack * pq, int operand);
21
22 // "Inserts" an operator into the stack and operates.
23 // Returns true if everything OK or false in other case.
24 int pushOperator(T_Stack * pq, char operator);
25
26 // Puts into data the number on top of the stack, and removes the top.
27 // Returns true if everything OK or false in other case.
28 int pop(T_Stack * pq, int * data);
29
30 // Frees the memory of a stack and sets it to empty.
31 void destroy(T_Stack * pq);
32
33 #endif /* STACK_H_ */
34
35
36 //TERMINA AQUI
```

```
1  //EMPIEZA AQUI
2
3
4  #ifndef STACK_H_
5  #define STACK_H_
6
7  typedef struct Node * T_Stack;
8  typedef struct Node {
9    struct Node * next;
10   int number;
11 } T_Node;
12
13 // Creates an empty stack.
14 T_Stack create();
15
16 // Returns true if the stack is empty and false in other case.
17 int isEmpty(T_Stack q);
18
19 // Inserts a number into the stack.
20 void push(T_Stack * pq, int operand);
21
22 // "Inserts" an operator into the stack and operates.
23 // Returns true if everything OK or false in other case.
24 int pushOperator(T_Stack * pq, char operator);
25
26 // Puts into data the number on top of the stack, and removes the top.
27 // Returns true if everything OK or false in other case.
28 int pop(T_Stack * pq, int * data);
29
30 // Frees the memory of a stack and sets it to empty.
31 void destroy(T_Stack * pq);
32
33 #endif /* STACK_H_ */
34
35
36 //TERMINA AQUI
```

```
1  //EMPIEZA AQUI
2
3
4  #ifndef STACK_H_
5  #define STACK_H_
6
7  typedef struct Node * T_Stack;
8  typedef struct Node {
9    struct Node * next;
10   int number;
11 } T_Node;
12
13 // Creates an empty stack.
14 T_Stack create();
15
16 // Returns true if the stack is empty and false in other case.
17 int isEmpty(T_Stack q);
18
19 // Inserts a number into the stack.
20 void push(T_Stack * pq, int operand);
21
22 // "Inserts" an operator into the stack and operates.
23 // Returns true if everything OK or false in other case.
24 int pushOperator(T_Stack * pq, char operator);
25
26 // Puts into data the number on top of the stack, and removes the top.
27 // Returns true if everything OK or false in other case.
28 int pop(T_Stack * pq, int * data);
29
30 // Frees the memory of a stack and sets it to empty.
31 void destroy(T_Stack * pq);
32
33 #endif /* STACK_H_ */
34
35
36 //TERMINA AQUI
```

```
1  //EMPIEZA AQUI
2
3
4  #ifndef STACK_H_
5  #define STACK_H_
6
7  typedef struct Node * T_Stack;
8  typedef struct Node {
9    struct Node * next;
10   int number;
11 } T_Node;
12
13 // Creates an empty stack.
14 T_Stack create();
15
16 // Returns true if the stack is empty and false in other case.
17 int isEmpty(T_Stack q);
18
19 // Inserts a number into the stack.
20 void push(T_Stack * pq, int operand);
21
22 // "Inserts" an operator into the stack and operates.
23 // Returns true if everything OK or false in other case.
24 int pushOperator(T_Stack * pq, char operator);
25
26 // Puts into data the number on top of the stack, and removes the top.
27 // Returns true if everything OK or false in other case.
28 int pop(T_Stack * pq, int * data);
29
30 // Frees the memory of a stack and sets it to empty.
31 void destroy(T_Stack * pq);
32
33 #endif /* STACK_H_ */
34
35
36 //TERMINA AQUI
```

```
1  //EMPIEZA AQUI
2
3
4  #ifndef STACK_H_
5  #define STACK_H_
6
7  typedef struct Node * T_Stack;
8  typedef struct Node {
9    struct Node * next;
10   int number;
11 } T_Node;
12
13 // Creates an empty stack.
14 T_Stack create();
15
16 // Returns true if the stack is empty and false in other case.
17 int isEmpty(T_Stack q);
18
19 // Inserts a number into the stack.
20 void push(T_Stack * pq, int operand);
21
22 // "Inserts" an operator into the stack and operates.
23 // Returns true if everything OK or false in other case.
24 int pushOperator(T_Stack * pq, char operator);
25
26 // Puts into data the number on top of the stack, and removes the top.
27 // Returns true if everything OK or false in other case.
28 int pop(T_Stack * pq, int * data);
29
30 // Frees the memory of a stack and sets it to empty.
31 void destroy(T_Stack * pq);
32
33 #endif /* STACK_H_ */
34
35
36 //TERMINA AQUI
```

```
1  //EMPIEZA AQUI
2
3
4  #ifndef STACK_H_
5  #define STACK_H_
6
7  typedef struct Node * T_Stack;
8  typedef struct Node {
9    struct Node * next;
10   int number;
11 } T_Node;
12
13 // Creates an empty stack.
14 T_Stack create();
15
16 // Returns true if the stack is empty and false in other case.
17 int isEmpty(T_Stack q);
18
19 // Inserts a number into the stack.
20 void push(T_Stack * pq, int operand);
21
22 // "Inserts" an operator into the stack and operates.
23 // Returns true if everything OK or false in other case.
24 int pushOperator(T_Stack * pq, char operator);
25
26 // Puts into data the number on top of the stack, and removes the top.
27 // Returns true if everything OK or false in other case.
28 int pop(T_Stack * pq, int * data);
29
30 // Frees the memory of a stack and sets it to empty.
31 void destroy(T_Stack * pq);
32
33 #endif /* STACK_H_ */
34
35
36 //TERMINA AQUI
```

```
1  //EMPIEZA AQUI
2
3
4  #ifndef STACK_H_
5  #define STACK_H_
6
7  typedef struct Node * T_Stack;
8  typedef struct Node {
9    struct Node * next;
10   int number;
11 } T_Node;
12
13 // Creates an empty stack.
14 T_Stack create();
15
16 // Returns true if the stack is empty and false in other case.
17 int isEmpty(T_Stack q);
18
19 // Inserts a number into the stack.
20 void push(T_Stack * pq, int operand);
21
22 // "Inserts" an operator into the stack and operates.
23 // Returns true if everything OK or false in other case.
24 int pushOperator(T_Stack * pq, char operator);
25
26 // Puts into data the number on top of the stack, and removes the top.
27 // Returns true if everything OK or false in other case.
28 int pop(T_Stack * pq, int * data);
29
30 // Frees the memory of a stack and sets it to empty.
31 void destroy(T_Stack * pq);
32
33 #endif /* STACK_H_ */
34
35
36 //TERMINA AQUI
```

```
1  //EMPIEZA AQUI
2
3
4  #ifndef STACK_H_
5  #define STACK_H_
6
7  typedef struct Node * T_Stack;
8  typedef struct Node {
9    struct Node * next;
10   int number;
11 } T_Node;
12
13 // Creates an empty stack.
14 T_Stack create();
15
16 // Returns true if the stack is empty and false in other case.
17 int isEmpty(T_Stack q);
18
19 // Inserts a number into the stack.
20 void push(T_Stack * pq, int operand);
21
22 // "Inserts" an operator into the stack and operates.
23 // Returns true if everything OK or false in other case.
24 int pushOperator(T_Stack * pq, char operator);
25
26 // Puts into data the number on top of the stack, and removes the top.
27 // Returns true if everything OK or false in other case.
28 int pop(T_Stack * pq, int * data);
29
30 // Frees the memory of a stack and sets it to empty.
31 void destroy(T_Stack * pq);
32
33 #endif /* STACK_H_ */
34
35
36 //TERMINA AQUI
```

```
1  //EMPIEZA AQUI
2
3
4  #ifndef STACK_H_
5  #define STACK_H_
6
7  typedef struct Node * T_Stack;
8  typedef struct Node {
9    struct Node * next;
10   int number;
11 } T_Node;
12
13 // Creates an empty stack.
14 T_Stack create();
15
16 // Returns true if the stack is empty and false in other case.
17 int isEmpty(T_Stack q);
18
19 // Inserts a number into the stack.
20 void push(T_Stack * pq, int operand);
21
22 // "Inserts" an operator into the stack and operates.
23 // Returns true if everything OK or false in other case.
24 int pushOperator(T_Stack * pq, char operator);
25
26 // Puts into data the number on top of the stack, and removes the top.
27 // Returns true if everything OK or false in other case.
28 int pop(T_Stack * pq, int * data);
29
30 // Frees the memory of a stack and sets it to empty.
31 void destroy(T_Stack * pq);
32
33 #endif /* STACK_H_ */
34
35
36 //TERMINA AQUI
```

```
1  //EMPIEZA AQUI
2
3
4  #ifndef STACK_H_
5  #define STACK_H_
6
7  typedef struct Node * T_Stack;
8  typedef struct Node {
9    struct Node * next;
10   int number;
11 } T_Node;
12
13 // Creates an empty stack.
14 T_Stack create();
15
16 // Returns true if the stack is empty and false in other case.
17 int isEmpty(T_Stack q);
18
19 // Inserts a number into the stack.
20 void push(T_Stack * pq, int operand);
21
22 // "Inserts" an operator into the stack and operates.
23 // Returns true if everything OK or false in other case.
24 int pushOperator(T_Stack * pq, char operator);
25
26 // Puts into data the number on top of the stack, and removes the top.
27 // Returns true if everything OK or false in other case.
28 int pop(T_Stack * pq, int * data);
29
30 // Frees the memory of a stack and sets it to empty.
31 void destroy(T_Stack * pq);
32
33 #endif /* STACK_H_ */
34
35
36 //TERMINA AQUI
```

```
1  //EMPIEZA AQUI
2
3
4  #ifndef STACK_H_
5  #define STACK_H_
6
7  typedef struct Node * T_Stack;
8  typedef struct Node {
9    struct Node * next;
10   int number;
11 } T_Node;
12
13 // Creates an empty stack.
14 T_Stack create();
15
16 // Returns true if the stack is empty and false in other case.
17 int isEmpty(T_Stack q);
18
19 // Inserts a number into the stack.
20 void push(T_Stack * pq, int operand);
21
22 // "Inserts" an operator into the stack and operates.
23 // Returns true if everything OK or false in other case.
24 int pushOperator(T_Stack * pq, char operator);
25
26 // Puts into data the number on top of the stack, and removes the top.
27 // Returns true if everything OK or false in other case.
28 int pop(T_Stack * pq, int * data);
29
30 // Frees the memory of a stack and sets it to empty.
31 void destroy(T_Stack * pq);
32
33 #endif /* STACK_H_ */
34
35
36 //TERMINA AQUI
```

```
1  //EMPIEZA AQUI
2
3
4  #ifndef STACK_H_
5  #define STACK_H_
6
7  typedef struct Node * T_Stack;
8  typedef struct Node {
9    struct Node * next;
10   int number;
11 } T_Node;
12
13 // Creates an empty stack.
14 T_Stack create();
15
16 // Returns true if the stack is empty and false in other case.
17 int isEmpty(T_Stack q);
18
19 // Inserts a number into the stack.
20 void push(T_Stack * pq, int operand);
21
22 // "Inserts" an operator into the stack and operates.
23 // Returns true if everything OK or false in other case.
24 int pushOperator(T_Stack * pq, char operator);
25
26 // Puts into data the number on top of the stack, and removes the top.
27 // Returns true if everything OK or false in other case.
28 int pop(T_Stack * pq, int * data);
29
30 // Frees the memory of a stack and sets it to empty.
31 void destroy(T_Stack * pq);
32
33 #endif /* STACK_H_ */
34
35
36 //TERMINA AQUI
```

```
1  //EMPIEZA AQUI
2
3
4  #ifndef STACK_H_
5  #define STACK_H_
6
7  typedef struct Node * T_Stack;
8  typedef struct Node {
9    struct Node * next;
10   int number;
11 } T_Node;
12
13 // Creates an empty stack.
14 T_Stack create();
15
16 // Returns true if the stack is empty and false in other case.
17 int isEmpty(T_Stack q);
18
19 // Inserts a number into the stack.
20 void push(T_Stack * pq, int operand);
21
22 // "Inserts" an operator into the stack and operates.
23 // Returns true if everything OK or false in other case.
24 int pushOperator(T_Stack * pq, char operator);
25
26 // Puts into data the number on top of the stack, and removes the top.
27 // Returns true if everything OK or false in other case.
28 int pop(T_Stack * pq, int * data);
29
30 // Frees the memory of a stack and sets it to empty.
31 void destroy(T_Stack * pq);
32
33 #endif /* STACK_H_ */
34
35
36 //TERMINA AQUI
```

```
1  //EMPIEZA AQUI
2
3
4  #ifndef STACK_H_
5  #define STACK_H_
6
7  typedef struct Node * T_Stack;
8  typedef struct Node {
9    struct Node * next;
10   int number;
11 } T_Node;
12
13 // Creates an empty stack.
14 T_Stack create();
15
16 // Returns true if the stack is empty and false in other case.
17 int isEmpty(T_Stack q);
18
19 // Inserts a number into the stack.
20 void push(T_Stack * pq, int operand);
21
22 // "Inserts" an operator into the stack and operates.
23 // Returns true if everything OK or false in other case.
24 int pushOperator(T_Stack * pq, char operator);
25
26 // Puts into data the number on top of the stack, and removes the top.
27 // Returns true if everything OK or false in other case.
28 int pop(T_Stack * pq, int * data);
29
30 // Frees the memory of a stack and sets it to empty.
31 void destroy(T_Stack * pq);
32
33 #endif /* STACK_H_ */
34
35
36 //TERMINA AQUI
```

```
1  //EMPIEZA AQUI
2
3
4  #ifndef STACK_H_
5  #define STACK_H_
6
7  typedef struct Node * T_Stack;
8  typedef struct Node {
9    struct Node * next;
10   int number;
11 } T_Node;
12
13 // Creates an empty stack.
14 T_Stack create();
15
16 // Returns true if the stack is empty and false in other case.
17 int isEmpty(T_Stack q);
18
19 // Inserts a number into the stack.
20 void push(T_Stack * pq, int operand);
21
22 // "Inserts" an operator into the stack and operates.
23 // Returns true if everything OK or false in other case.
24 int pushOperator(T_Stack * pq, char operator);
25
26 // Puts into data the number on top of the stack, and removes the top.
27 // Returns true if everything OK or false in other case.
28 int pop(T_Stack * pq, int * data);
29
30 // Frees the memory of a stack and sets it to empty.
31 void destroy(T_Stack * pq);
32
33 #endif /* STACK_H_ */
34
35
36 //TERMINA AQUI
```

```
1  //EMPIEZA AQUI
2
3
4  #ifndef STACK_H_
5  #define STACK_H_
6
7  typedef struct Node * T_Stack;
8  typedef struct Node {
9    struct Node * next;
10   int number;
11 } T_Node;
12
13 // Creates an empty stack.
14 T_Stack create();
15
16 // Returns true if the stack is empty and false in other case.
17 int isEmpty(T_Stack q);
18
19 // Inserts a number into the stack.
20 void push(T_Stack * pq, int operand);
21
22 // "Inserts" an operator into the stack and operates.
23 // Returns true if everything OK or false in other case.
24 int pushOperator(T_Stack * pq, char operator);
25
26 // Puts into data the number on top of the stack, and removes the top.
27 // Returns true if everything OK or false in other case.
28 int pop(T_Stack * pq, int * data);
29
30 // Frees the memory of a stack and sets it to empty.
31 void destroy(T_Stack * pq);
32
33 #endif /* STACK_H_ */
34
35
36 //TERMINA AQUI
```

```
1  //EMPIEZA AQUI
2
3
4  #ifndef STACK_H_
5  #define STACK_H_
6
7  typedef struct Node * T_Stack;
8  typedef struct Node {
9    struct Node * next;
10   int number;
11 } T_Node;
12
13 // Creates an empty stack.
14 T_Stack create();
15
16 // Returns true if the stack is empty and false in other case.
17 int isEmpty(T_Stack q);
18
19 // Inserts a number into the stack.
20 void push(T_Stack * pq, int operand);
21
22 // "Inserts" an operator into the stack and operates.
23 // Returns true if everything OK or false in other case.
24 int pushOperator(T_Stack * pq, char operator);
25
26 // Puts into data the number on top of the stack, and removes the top.
27 // Returns true if everything OK or false in other case.
28 int pop(T_Stack * pq, int * data);
29
30 // Frees the memory of a stack and sets it to empty.
31 void destroy(T_Stack * pq);
32
33 #endif /* STACK_H_ */
34
35
36 //TERMINA AQUI
```

```
1  //EMPIEZA AQUI
2
3
4  #ifndef STACK_H_
5  #define STACK_H_
6
7  typedef struct Node * T_Stack;
8  typedef struct Node {
9    struct Node * next;
10   int number;
11 } T_Node;
12
13 // Creates an empty stack.
14 T_Stack create();
15
16 // Returns true if the stack is empty and false in other case.
17 int isEmpty(T_Stack q);
18
19 // Inserts a number into the stack.
20 void push(T_Stack * pq, int operand);
21
22 // "Inserts" an operator into the stack and operates.
23 // Returns true if everything OK or false in other case.
24 int pushOperator(T_Stack * pq, char operator);
25
26 // Puts into data the number on top of the stack, and removes the top.
27 // Returns true if everything OK or false in other case.
28 int pop(T_Stack * pq, int * data);
29
30 // Frees the memory of a stack and sets it to empty.
31 void destroy(T_Stack * pq);
32
33 #endif /* STACK_H_ */
34
35
36 //TERMINA AQUI
```

```
1  //EMPIEZA AQUI
2
3
4  #ifndef STACK_H_
5  #define STACK_H_
6
7  typedef struct Node * T_Stack;
8  typedef struct Node {
9    struct Node * next;
10   int number;
11 } T_Node;
12
13 // Creates an empty stack.
14 T_Stack create();
15
16 // Returns true if the stack is empty and false in other case.
17 int isEmpty(T_Stack q);
18
19 // Inserts a number into the stack.
20 void push(T_Stack * pq, int operand);
21
22 // "Inserts" an operator into the stack and operates.
23 // Returns true if everything OK or false in other case.
24 int pushOperator(T_Stack * pq, char operator);
25
26 // Puts into data the number on top of the stack, and removes the top.
27 // Returns true if everything OK or false in other case.
28 int pop(T_Stack * pq, int * data);
29
30 // Frees the memory of a stack and sets it to empty.
31 void destroy(T_Stack * pq);
32
33 #endif /* STACK_H_ */
34
35
36 //TERMINA AQUI
```

```
1  //EMPIEZA AQUI
2
3
4  #ifndef STACK_H_
5  #define STACK_H_
6
7  typedef struct Node * T_Stack;
8  typedef struct Node {
9    struct Node * next;
10   int number;
11 } T_Node;
12
13 // Creates an empty stack.
14 T_Stack create();
15
16 // Returns true if the stack is empty and false in other case.
17 int isEmpty(T_Stack q);
18
19 // Inserts a number into the stack.
20 void push(T_Stack * pq, int operand);
21
22 // "Inserts" an operator into the stack and operates.
23 // Returns true if everything OK or false in other case.
24 int pushOperator(T_Stack * pq, char operator);
25
26 // Puts into data the number on top of the stack, and removes the top.
27 // Returns true if everything OK or false in other case.
28 int pop(T_Stack * pq, int * data);
29
30 // Frees the memory of a stack and sets it to empty.
31 void destroy(T_Stack * pq);
32
33 #endif /* STACK_H_ */
34
35
36 //TERMINA AQUI
```

```
1  //EMPIEZA AQUI
2
3
4  #ifndef STACK_H_
5  #define STACK_H_
6
7  typedef struct Node * T_Stack;
8  typedef struct Node {
9    struct Node * next;
10   int number;
11 } T_Node;
12
13 // Creates an empty stack.
14 T_Stack create();
15
16 // Returns true if the stack is empty and false in other case.
17 int isEmpty(T_Stack q);
18
19 // Inserts a number into the stack.
20 void push(T_Stack * pq, int operand);
21
22 // "Inserts" an operator into the stack and operates.
23 // Returns true if everything OK or false in other case.
24 int pushOperator(T_Stack * pq, char operator);
25
26 // Puts into data the number on top of the stack, and removes the top.
27 // Returns true if everything OK or false in other case.
28 int pop(T_Stack * pq, int * data);
29
30 // Frees the memory of a stack and sets it to empty.
31 void destroy(T_Stack * pq);
32
33 #endif /* STACK_H_ */
34
35
36 //TERMINA AQUI
```

```
1  //EMPIEZA AQUI
2
3
4  #ifndef STACK_H_
5  #define STACK_H_
6
7  typedef struct Node * T_Stack;
8  typedef struct Node {
9    struct Node * next;
10   int number;
11 } T_Node;
12
13 // Creates an empty stack.
14 T_Stack create();
15
16 // Returns true if the stack is empty and false in other case.
17 int isEmpty(T_Stack q);
18
19 // Inserts a number into the stack.
20 void push(T_Stack * pq, int operand);
21
22 // "Inserts" an operator into the stack and operates.
23 // Returns true if everything OK or false in other case.
24 int pushOperator(T_Stack * pq, char operator);
25
26 // Puts into data the number on top of the stack, and removes the top.
27 // Returns true if everything OK or false in other case.
28 int pop(T_Stack * pq, int * data);
29
30 // Frees the memory of a stack and sets it to empty.
31 void destroy(T_Stack * pq);
32
33 #endif /* STACK_H_ */
34
35
36 //TERMINA AQUI
```

```
1  //EMPIEZA AQUI
2
3
4  #ifndef STACK_H_
5  #define STACK_H_
6
7  typedef struct Node * T_Stack;
8  typedef struct Node {
9    struct Node * next;
10   int number;
11 } T_Node;
12
13 // Creates an empty stack.
14 T_Stack create();
15
16 // Returns true if the stack is empty and false in other case.
17 int isEmpty(T_Stack q);
18
19 // Inserts a number into the stack.
20 void push(T_Stack * pq, int operand);
21
22 // "Inserts" an operator into the stack and operates.
23 // Returns true if everything OK or false in other case.
24 int pushOperator(T_Stack * pq, char operator);
25
26 // Puts into data the number on top of the stack, and removes the top.
27 // Returns true if everything OK or false in other case.
28 int pop(T_Stack * pq, int * data);
29
30 // Frees the memory of a stack and sets it to empty.
31 void destroy(T_Stack * pq);
32
33 #endif /* STACK_H_ */
34
35
36 //TERMINA AQUI
```

```
1  //EMPIEZA AQUI
2
3
4  #ifndef STACK_H_
5  #define STACK_H_
6
7  typedef struct Node * T_Stack;
8  typedef struct Node {
9    struct Node * next;
10   int number;
11 } T_Node;
12
13 // Creates an empty stack.
14 T_Stack create();
15
16 // Returns true if the stack is empty and false in other case.
17 int isEmpty(T_Stack q);
18
19 // Inserts a number into the stack.
20 void push(T_Stack * pq, int operand);
21
22 // "Inserts" an operator into the stack and operates.
23 // Returns true if everything OK or false in other case.
24 int pushOperator(T_Stack * pq, char operator);
25
26 // Puts into data the number on top of the stack, and removes the top.
27 // Returns true if everything OK or false in other case.
28 int pop(T_Stack * pq, int * data);
29
30 // Frees the memory of a stack and sets it to empty.
31 void destroy(T_Stack * pq);
32
33 #endif /* STACK_H_ */
34
35
36 //TERMINA AQUI
```

```
1  //EMPIEZA AQUI
2
3
4  #ifndef STACK_H_
5  #define STACK_H_
6
7  typedef struct Node * T_Stack;
8  typedef struct Node {
9    struct Node * next;
10   int number;
11 } T_Node;
12
13 // Creates an empty stack.
14 T_Stack create();
15
16 // Returns true if the stack is empty and false in other case.
17 int isEmpty(T_Stack q);
18
19 // Inserts a number into the stack.
20 void push(T_Stack * pq, int operand);
21
22 // "Inserts" an operator into the stack and operates.
23 // Returns true if everything OK or false in other case.
24 int pushOperator(T_Stack * pq, char operator);
25
26 // Puts into data the number on top of the stack, and removes the top.
27 // Returns true if everything OK or false in other case.
28 int pop(T_Stack * pq, int * data);
29
30 // Frees the memory of a stack and sets it to empty.
31 void destroy(T_Stack * pq);
32
33 #endif /* STACK_H_ */
34
35
36 //TERMINA AQUI
```

```
1  //EMPIEZA AQUI
2
3
4  #ifndef STACK_H_
5  #define STACK_H_
6
7  typedef struct Node * T_Stack;
8  typedef struct Node {
9    struct Node * next;
10   int number;
11 } T_Node;
12
13 // Creates an empty stack.
14 T_Stack create();
15
16 // Returns true if the stack is empty and false in other case.
17 int isEmpty(T_Stack q);
18
19 // Inserts a number into the stack.
20 void push(T_Stack * pq, int operand);
21
22 // "Inserts" an operator into the stack and operates.
23 // Returns true if everything OK or false in other case.
24 int pushOperator(T_Stack * pq, char operator);
25
26 // Puts into data the number on top of the stack, and removes the top.
27 // Returns true if everything OK or false in other case.
28 int pop(T_Stack * pq, int * data);
29
30 // Frees the memory of a stack and sets it to empty.
31 void destroy(T_Stack * pq);
32
33 #endif /* STACK_H_ */
34
35
36 //TERMINA AQUI
```

```
1  //EMPIEZA AQUI
2
3
4  #ifndef STACK_H_
5  #define STACK_H_
6
7  typedef struct Node * T_Stack;
8  typedef struct Node {
9    struct Node * next;
10   int number;
11 } T_Node;
12
13 // Creates an empty stack.
14 T_Stack create();
15
16 // Returns true if the stack is empty and false in other case.
17 int isEmpty(T_Stack q);
18
19 // Inserts a number into the stack.
20 void push(T_Stack * pq, int operand);
21
22 // "Inserts" an operator into the stack and operates.
23 // Returns true if everything OK or false in other case.
24 int pushOperator(T_Stack * pq, char operator);
25
26 // Puts into data the number on top of the stack, and removes the top.
27 // Returns true if everything OK or false in other case.
28 int pop(T_Stack * pq, int * data);
29
30 // Frees the memory of a stack and sets it to empty.
31 void destroy(T_Stack * pq);
32
33 #endif /* STACK_H_ */
34
35
36 //TERMINA AQUI
```

```
1  //EMPIEZA AQUI
2
3
4  #ifndef STACK_H_
5  #define STACK_H_
6
7  typedef struct Node * T_Stack;
8  typedef struct Node {
9    struct Node * next;
10   int number;
11 } T_Node;
12
13 // Creates an empty stack.
14 T_Stack create();
15
16 // Returns true if the stack is empty and false in other case.
17 int isEmpty(T_Stack q);
18
19 // Inserts a number into the stack.
20 void push(T_Stack * pq, int operand);
21
22 // "Inserts" an operator into the stack and operates.
23 // Returns true if everything OK or false in other case.
24 int pushOperator(T_Stack * pq, char operator);
25
26 // Puts into data the number on top of the stack, and removes the top.
27 // Returns true if everything OK or false in other case.
28 int pop(T_Stack * pq, int * data);
29
30 // Frees the memory of a stack and sets it to empty.
31 void destroy(T_Stack * pq);
32
33 #endif /* STACK_H_ */
34
35
36 //TERMINA AQUI
```

```
1  //EMPIEZA AQUI
2
3
4  #ifndef STACK_H_
5  #define STACK_H_
6
7  typedef struct Node * T_Stack;
8  typedef struct Node {
9    struct Node * next;
10   int number;
11 } T_Node;
12
13 // Creates an empty stack.
14 T_Stack create();
15
16 // Returns true if the stack is empty and false in other case.
17 int isEmpty(T_Stack q);
18
19 // Inserts a number into the stack.
20 void push(T_Stack * pq, int operand);
21
22 // "Inserts" an operator into the stack and operates.
23 // Returns true if everything OK or false in other case.
24 int pushOperator(T_Stack * pq, char operator);
25
26 // Puts into data the number on top of the stack, and removes the top.
27 // Returns true if everything OK or false in other case.
28 int pop(T_Stack * pq, int * data);
29
30 // Frees the memory of a stack and sets it to empty.
31 void destroy(T_Stack * pq);
32
33 #endif /* STACK_H_ */
34
35
36 //TERMINA AQUI
```

```
1  //EMPIEZA AQUI
2
3
4  #ifndef STACK_H_
5  #define STACK_H_
6
7  typedef struct Node * T_Stack;
8  typedef struct Node {
9    struct Node * next;
10   int number;
11 } T_Node;
12
13 // Creates an empty stack.
14 T_Stack create();
15
16 // Returns true if the stack is empty and false in other case.
17 int isEmpty(T_Stack q);
18
19 // Inserts a number into the stack.
20 void push(T_Stack * pq, int operand);
21
22 // "Inserts" an operator into the stack and operates.
23 // Returns true if everything OK or false in other case.
24 int pushOperator(T_Stack * pq, char operator);
25
26 // Puts into data the number on top of the stack, and removes the top.
27 // Returns true if everything OK or false in other case.
28 int pop(T_Stack * pq, int * data);
29
30 // Frees the memory of a stack and sets it to empty.
31 void destroy(T_Stack * pq);
32
33 #endif /* STACK_H_ */
34
35
36 //TERMINA AQUI
```

```
1  //EMPIEZA AQUI
2
3
4  #ifndef STACK_H_
5  #define STACK_H_
6
7  typedef struct Node * T_Stack;
8  typedef struct Node {
9    struct Node * next;
10   int number;
11 } T_Node;
12
13 // Creates an empty stack.
14 T_Stack create();
15
16 // Returns true if the stack is empty and false in other case.
17 int isEmpty(T_Stack q);
18
19 // Inserts a number into the stack.
20 void push(T_Stack * pq, int operand);
21
22 // "Inserts" an operator into the stack and operates.
23 // Returns true if everything OK or false in other case.
24 int pushOperator(T_Stack * pq, char operator);
25
26 // Puts into data the number on top of the stack, and removes the top.
27 // Returns true if everything OK or false in other case.
28 int pop(T_Stack * pq, int * data);
29
30 // Frees the memory of a stack and sets it to empty.
31 void destroy(T_Stack * pq);
32
33 #endif /* STACK_H_ */
34
35
36 //TERMINA AQUI
```

```
1  //EMPIEZA AQUI
2
3
4  #ifndef STACK_H_
5  #define STACK_H_
6
7  typedef struct Node * T_Stack;
8  typedef struct Node {
9    struct Node * next;
10   int number;
11 } T_Node;
12
13 // Creates an empty stack.
14 T_Stack create();
15
16 // Returns true if the stack is empty and false in other case.
17 int isEmpty(T_Stack q);
18
19 // Inserts a number into the stack.
20 void push(T_Stack * pq, int operand);
21
22 // "Inserts" an operator into the stack and operates.
23 // Returns true if everything OK or false in other case.
24 int pushOperator(T_Stack * pq, char operator);
25
26 // Puts into data the number on top of the stack, and removes the top.
27 // Returns true if everything OK or false in other case.
28 int pop(T_Stack * pq, int * data);
29
30 // Frees the memory of a stack and sets it to empty.
31 void destroy(T_Stack * pq);
32
33 #endif /* STACK_H_ */
34
35
36 //TERMINA AQUI
```

```
1  //EMPIEZA AQUI
2
3
4  #ifndef STACK_H_
5  #define STACK_H_
6
7  typedef struct Node * T_Stack;
8  typedef struct Node {
9    struct Node * next;
10   int number;
11 } T_Node;
12
13 // Creates an empty stack.
14 T_Stack create();
15
16 // Returns true if the stack is empty and false in other case.
17 int isEmpty(T_Stack q);
18
19 // Inserts a number into the stack.
20 void push(T_Stack * pq, int operand);
21
22 // "Inserts" an operator into the stack and operates.
23 // Returns true if everything OK or false in other case.
24 int pushOperator(T_Stack * pq, char operator);
25
26 // Puts into data the number on top of the stack, and removes the top.
27 // Returns true if everything OK or false in other case.
28 int pop(T_Stack * pq, int * data);
29
30 // Frees the memory of a stack and sets it to empty.
31 void destroy(T_Stack * pq);
32
33 #endif /* STACK_H_ */
34
35
36 //TERMINA AQUI
```

```
1  //EMPIEZA AQUI
2
3
4  #ifndef STACK_H_
5  #define STACK_H_
6
7  typedef struct Node * T_Stack;
8  typedef struct Node {
9    struct Node * next;
10   int number;
11 } T_Node;
12
13 // Creates an empty stack.
14 T_Stack create();
15
16 // Returns true if the stack is empty and false in other case.
17 int isEmpty(T_Stack q);
18
19 // Inserts a number into the stack.
20 void push(T_Stack * pq, int operand);
21
22 // "Inserts" an operator into the stack and operates.
23 // Returns true if everything OK or false in other case.
24 int pushOperator(T_Stack * pq, char operator);
25
26 // Puts into data the number on top of the stack, and removes the top.
27 // Returns true if everything OK or false in other case.
28 int pop(T_Stack * pq, int * data);
29
30 // Frees the memory of a stack and sets it to empty.
31 void destroy(T_Stack * pq);
32
33 #endif /* STACK_H_ */
34
35
36 //TERMINA AQUI
```

```
1  //EMPIEZA AQUI
2
3
4  #ifndef STACK_H_
5  #define STACK_H_
6
7  typedef struct Node * T_Stack;
8  typedef struct Node {
9    struct Node * next;
10   int number;
11 } T_Node;
12
13 // Creates an empty stack.
14 T_Stack create();
15
16 // Returns true if the stack is empty and false in other case.
17 int isEmpty(T_Stack q);
18
19 // Inserts a number into the stack.
20 void push(T_Stack * pq, int operand);
21
22 // "Inserts" an operator into the stack and operates.
23 // Returns true if everything OK or false in other case.
24 int pushOperator(T_Stack * pq, char operator);
25
26 // Puts into data the number on top of the stack, and removes the top.
27 // Returns true if everything OK or false in other case.
28 int pop(T_Stack * pq, int * data);
29
30 // Frees the memory of a stack and sets it to empty.
31 void destroy(T_Stack * pq);
32
33 #endif /* STACK_H_ */
34
35
36 //TERMINA AQUI
```

```
1  //EMPIEZA AQUI
2
3
4  #ifndef STACK_H_
5  #define STACK_H_
6
7  typedef struct Node * T_Stack;
8  typedef struct Node {
9    struct Node * next;
10   int number;
11 } T_Node;
12
13 // Creates an empty stack.
14 T_Stack create();
15
16 // Returns true if the stack is empty and false in other case.
17 int isEmpty(T_Stack q);
18
19 // Inserts a number into the stack.
20 void push(T_Stack * pq, int operand);
21
22 // "Inserts" an operator into the stack and operates.
23 // Returns true if everything OK or false in other case.
24 int pushOperator(T_Stack * pq, char operator);
25
26 // Puts into data the number on top of the stack, and removes the top.
27 // Returns true if everything OK or false in other case.
28 int pop(T_Stack * pq, int * data);
29
30 // Frees the memory of a stack and sets it to empty.
31 void destroy(T_Stack * pq);
32
33 #endif /* STACK_H_ */
34
35
36 //TERMINA AQUI
```

```
1  //EMPIEZA AQUI
2
3
4  #ifndef STACK_H_
5  #define STACK_H_
6
7  typedef struct Node * T_Stack;
8  typedef struct Node {
9    struct Node * next;
10   int number;
11 } T_Node;
12
13 // Creates an empty stack.
14 T_Stack create();
15
16 // Returns true if the stack is empty and false in other case.
17 int isEmpty(T_Stack q);
18
19 // Inserts a number into the stack.
20 void push(T_Stack * pq, int operand);
21
22 // "Inserts" an operator into the stack and operates.
23 // Returns true if everything OK or false in other case.
24 int pushOperator(T_Stack * pq, char operator);
25
26 // Puts into data the number on top of the stack, and removes the top.
27 // Returns true if everything OK or false in other case.
28 int pop(T_Stack * pq, int * data);
29
30 // Frees the memory of a stack and sets it to empty.
31 void destroy(T_Stack * pq);
32
33 #endif /* STACK_H_ */
34
35
36 //TERMINA AQUI
```

```
1  //EMPIEZA AQUI
2
3
4  #ifndef STACK_H_
5  #define STACK_H_
6
7  typedef struct Node * T_Stack;
8  typedef struct Node {
9    struct Node * next;
10   int number;
11 } T_Node;
12
13 // Creates an empty stack.
14 T_Stack create();
15
16 // Returns true if the stack is empty and false in other case.
17 int isEmpty(T_Stack q);
18
19 // Inserts a number into the stack.
20 void push(T_Stack * pq, int operand);
21
22 // "Inserts" an operator into the stack and operates.
23 // Returns true if everything OK or false in other case.
24 int pushOperator(T_Stack * pq, char operator);
25
26 // Puts into data the number on top of the stack, and removes the top.
27 // Returns true if everything OK or false in other case.
28 int pop(T_Stack * pq, int * data);
29
30 // Frees the memory of a stack and sets it to empty.
31 void destroy(T_Stack * pq);
32
33 #endif /* STACK_H_ */
34
35
36 //TERMINA AQUI
```

```
1  //EMPIEZA AQUI
2
3
4  #ifndef STACK_H_
5  #define STACK_H_
6
7  typedef struct Node * T_Stack;
8  typedef struct Node {
9    struct Node * next;
10   int number;
11 } T_Node;
12
13 // Creates an empty stack.
14 T_Stack create();
15
16 // Returns true if the stack is empty and false in other case.
17 int isEmpty(T_Stack q);
18
19 // Inserts a number into the stack.
20 void push(T_Stack * pq, int operand);
21
22 // "Inserts" an operator into the stack and operates.
23 // Returns true if everything OK or false in other case.
24 int pushOperator(T_Stack * pq, char operator);
25
26 // Puts into data the number on top of the stack, and removes the top.
27 // Returns true if everything OK or false in other case.
28 int pop(T_Stack * pq, int * data);
29
30 // Frees the memory of a stack and sets it to empty.
31 void destroy(T_Stack * pq);
32
33 #endif /* STACK_H_ */
34
35
36 //TERMINA AQUI
```

```
1  //EMPIEZA AQUI
2
3
4  #ifndef STACK_H_
5  #define STACK_H_
6
7  typedef struct Node * T_Stack;
8  typedef struct Node {
9    struct Node * next;
10   int number;
11 } T_Node;
12
13 // Creates an empty stack.
14 T_Stack create();
15
16 // Returns true if the stack is empty and false in other case.
17 int isEmpty(T_Stack q);
18
19 // Inserts a number into the stack.
20 void push(T_Stack * pq, int operand);
21
22 // "Inserts" an operator into the stack and operates.
23 // Returns true if everything OK or false in other case.
24 int pushOperator(T_Stack * pq, char operator);
25
26 // Puts into data the number on top of the stack, and removes the top.
27 // Returns true if everything OK or false in other case.
28 int pop(T_Stack * pq, int * data);
29
30 // Frees the memory of a stack and sets it to empty.
31 void destroy(T_Stack * pq);
32
33 #endif /* STACK_H_ */
34
35
36 //TERMINA AQUI
```

```
1  //EMPIEZA AQUI
2
3
4  #ifndef STACK_H_
5  #define STACK_H_
6
7  typedef struct Node * T_Stack;
8  typedef struct Node {
9    struct Node * next;
10   int number;
11 } T_Node;
12
13 // Creates an empty stack.
14 T_Stack create();
15
16 // Returns true if the stack is empty and false in other case.
17 int isEmpty(T_Stack q);
18
19 // Inserts a number into the stack.
20 void push(T_Stack * pq, int operand);
21
22 // "Inserts" an operator into the stack and operates.
23 // Returns true if everything OK or false in other case.
24 int pushOperator(T_Stack * pq, char operator);
25
26 // Puts into data the number on top of the stack, and removes the top.
27 // Returns true if everything OK or false in other case.
28 int pop(T_Stack * pq, int * data);
29
30 // Frees the memory of a stack and sets it to empty.
31 void destroy(T_Stack * pq);
32
33 #endif /* STACK_H_ */
34
35
36 //TERMINA AQUI
```

```
1  //EMPIEZA AQUI
2
3
4  #ifndef STACK_H_
5  #define STACK_H_
6
7  typedef struct Node * T_Stack;
8  typedef struct Node {
9    struct Node * next;
10   int number;
11 } T_Node;
12
13 // Creates an empty stack.
14 T_Stack create();
15
16 // Returns true if the stack is empty and false in other case.
17 int isEmpty(T_Stack q);
18
19 // Inserts a number into the stack.
20 void push(T_Stack * pq, int operand);
21
22 // "Inserts" an operator into the stack and operates.
23 // Returns true if everything OK or false in other case.
24 int pushOperator(T_Stack * pq, char operator);
25
26 // Puts into data the number on top of the stack, and removes the top.
27 // Returns true if everything OK or false in other case.
28 int pop(T_Stack * pq, int * data);
29
30 // Frees the memory of a stack and sets it to empty.
31 void destroy(T_Stack * pq);
32
33 #endif /* STACK_H_ */
34
35
36 //TERMINA AQUI
```

```
1  //EMPIEZA AQUI
2
3
4  #ifndef STACK_H_
5  #define STACK_H_
6
7  typedef struct Node * T_Stack;
8  typedef struct Node {
9    struct Node * next;
10   int number;
11 } T_Node;
12
13 // Creates an empty stack.
14 T_Stack create();
15
16 // Returns true if the stack is empty and false in other case.
17 int isEmpty(T_Stack q);
18
19 // Inserts a number into the stack.
20 void push(T_Stack * pq, int operand);
21
22 // "Inserts" an operator into the stack and operates.
23 // Returns true if everything OK or false in other case.
24 int pushOperator(T_Stack * pq, char operator);
25
26 // Puts into data the number on top of the stack, and removes the top.
27 // Returns true if everything OK or false in other case.
28 int pop(T_Stack * pq, int * data);
29
30 // Frees the memory of a stack and sets it to empty.
31 void destroy(T_Stack * pq);
32
33 #endif /* STACK_H_ */
34
35
36 //TERMINA AQUI
```

```
1  //EMPIEZA AQUI
2
3
4  #ifndef STACK_H_
5  #define STACK_H_
6
7  typedef struct Node * T_Stack;
8  typedef struct Node {
9    struct Node * next;
10   int number;
11 } T_Node;
12
13 // Creates an empty stack.
14 T_Stack create();
15
16 // Returns true if the stack is empty and false in other case.
17 int isEmpty(T_Stack q);
18
19 // Inserts a number into the stack.
20 void push(T_Stack * pq, int operand);
21
22 // "Inserts" an operator into the stack and operates.
23 // Returns true if everything OK or false in other case.
24 int pushOperator(T_Stack * pq, char operator);
25
26 // Puts into data the number on top of the stack, and removes the top.
27 // Returns true if everything OK or false in other case.
28 int pop(T_Stack * pq, int * data);
29
30 // Frees the memory of a stack and sets it to empty.
31 void destroy(T_Stack * pq);
32
33 #endif /* STACK_H_ */
34
35
36 //TERMINA AQUI
```

```
1  //EMPIEZA AQUI
2
3
4  #ifndef STACK_H_
5  #define STACK_H_
6
7  typedef struct Node * T_Stack;
8  typedef struct Node {
9    struct Node * next;
10   int number;
11 } T_Node;
12
13 // Creates an empty stack.
14 T_Stack create();
15
16 // Returns true if the stack is empty and false in other case.
17 int isEmpty(T_Stack q);
18
19 // Inserts a number into the stack.
20 void push(T_Stack * pq, int operand);
21
22 // "Inserts" an operator into the stack and operates.
23 // Returns true if everything OK or false in other case.
24 int pushOperator(T_Stack * pq, char operator);
25
26 // Puts into data the number on top of the stack, and removes the top.
27 // Returns true if everything OK or false in other case.
28 int pop(T_Stack * pq, int * data);
29
30 // Frees the memory of a stack and sets it to empty.
31 void destroy(T_Stack * pq);
32
33 #endif /* STACK_H_ */
34
35
36 //TERMINA AQUI
```

```
1  //EMPIEZA AQUI
2
3
4  #ifndef STACK_H_
5  #define STACK_H_
6
7  typedef struct Node * T_Stack;
8  typedef struct Node {
9    struct Node * next;
10   int number;
11 } T_Node;
12
13 // Creates an empty stack.
14 T_Stack create();
15
16 // Returns true if the stack is empty and false in other case.
17 int isEmpty(T_Stack q);
18
19 // Inserts a number into the stack.
20 void push(T_Stack * pq, int operand);
21
22 // "Inserts" an operator into the stack and operates.
23 // Returns true if everything OK or false in other case.
24 int pushOperator(T_Stack * pq, char operator);
25
26 // Puts into data the number on top of the stack, and removes the top.
27 // Returns true if everything OK or false in other case.
28 int pop(T_Stack * pq, int * data);
29
30 // Frees the memory of a stack and sets it to empty.
31 void destroy(T_Stack * pq);
32
33 #endif /* STACK_H_ */
34
35
36 //TERMINA AQUI
```

```
1  //EMPIEZA AQUI
2
3
4  #ifndef STACK_H_
5  #define STACK_H_
6
7  typedef struct Node * T_Stack;
8  typedef struct Node {
9    struct Node * next;
10   int number;
11 } T_Node;
12
13 // Creates an empty stack.
14 T_Stack create();
15
16 // Returns true if the stack is empty and false in other case.
17 int isEmpty(T_Stack q);
18
19 // Inserts a number into the stack.
20 void push(T_Stack * pq, int operand);
21
22 // "Inserts" an operator into the stack and operates.
23 // Returns true if everything OK or false in other case.
24 int pushOperator(T_Stack * pq, char operator);
25
26 // Puts into data the number on top of the stack, and removes the top.
27 // Returns true if everything OK or false in other case.
28 int pop(T_Stack * pq, int * data);
29
30 // Frees the memory of a stack and sets it to empty.
31 void destroy(T_Stack * pq);
32
33 #endif /* STACK_H_ */
34
35
36 //TERMINA AQUI
```

```
1  //EMPIEZA AQUI
2
3
4  #ifndef STACK_H_
5  #define STACK_H_
6
7  typedef struct Node * T_Stack;
8  typedef struct Node {
9    struct Node * next;
10   int number;
11 } T_Node;
12
13 // Creates an empty stack.
14 T_Stack create();
15
16 // Returns true if the stack is empty and false in other case.
17 int isEmpty(T_Stack q);
18
19 // Inserts a number into the stack.
20 void push(T_Stack * pq, int operand);
21
22 // "Inserts" an operator into the stack and operates.
23 // Returns true if everything OK or false in other case.
24 int pushOperator(T_Stack * pq, char operator);
25
26 // Puts into data the number on top of the stack, and removes the top.
27 // Returns true if everything OK or false in other case.
28 int pop(T_Stack * pq, int * data);
29
30 // Frees the memory of a stack and sets it to empty.
31 void destroy(T_Stack * pq);
32
33 #endif /* STACK_H_ */
34
35
36 //TERMINA AQUI
```

```
1  //EMPIEZA AQUI
2
3
4  #ifndef STACK_H_
5  #define STACK_H_
6
7  typedef struct Node * T_Stack;
8  typedef struct Node {
9    struct Node * next;
10   int number;
11 } T_Node;
12
13 // Creates an empty stack.
14 T_Stack create();
15
16 // Returns true if the stack is empty and false in other case.
17 int isEmpty(T_Stack q);
18
19 // Inserts a number into the stack.
20 void push(T_Stack * pq, int operand);
21
22 // "Inserts" an operator into the stack and operates.
23 // Returns true if everything OK or false in other case.
24 int pushOperator(T_Stack * pq, char operator);
25
26 // Puts into data the number on top of the stack, and removes the top.
27 // Returns true if everything OK or false in other case.
28 int pop(T_Stack * pq, int * data);
29
30 // Frees the memory of a stack and sets it to empty.
31 void destroy(T_Stack * pq);
32
33 #endif /* STACK_H_ */
34
35
36 //TERMINA AQUI
```

```
1  //EMPIEZA AQUI
2
3
4  #ifndef STACK_H_
5  #define STACK_H_
6
7  typedef struct Node * T_Stack;
8  typedef struct Node {
9    struct Node * next;
10   int number;
11 } T_Node;
12
13 // Creates an empty stack.
14 T_Stack create();
15
16 // Returns true if the stack is empty and false in other case.
17 int isEmpty(T_Stack q);
18
19 // Inserts a number into the stack.
20 void push(T_Stack * pq, int operand);
21
22 // "Inserts" an operator into the stack and operates.
23 // Returns true if everything OK or false in other case.
24 int pushOperator(T_Stack * pq, char operator);
25
26 // Puts into data the number on top of the stack, and removes the top.
27 // Returns true if everything OK or false in other case.
28 int pop(T_Stack * pq, int * data);
29
30 // Frees the memory of a stack and sets it to empty.
31 void destroy(T_Stack * pq);
32
33 #endif /* STACK_H_ */
34
35
36 //TERMINA AQUI
```

```
1  //EMPIEZA AQUI
2
3
4  #ifndef STACK_H_
5  #define STACK_H_
6
7  typedef struct Node * T_Stack;
8  typedef struct Node {
9    struct Node * next;
10   int number;
11 } T_Node;
12
13 // Creates an empty stack.
14 T_Stack create();
15
16 // Returns true if the stack is empty and false in other case.
17 int isEmpty(T_Stack q);
18
19 // Inserts a number into the stack.
20 void push(T_Stack * pq, int operand);
21
22 // "Inserts" an operator into the stack and operates.
23 // Returns true if everything OK or false in other case.
24 int pushOperator(T_Stack * pq, char operator);
25
26 // Puts into data the number on top of the stack, and removes the top.
27 // Returns true if everything OK or false in other case.
28 int pop(T_Stack * pq, int * data);
29
30 // Frees the memory of a stack and sets it to empty.
31 void destroy(T_Stack * pq);
32
33 #endif /* STACK_H_ */
34
35
36 //TERMINA AQUI
```

```
1  //EMPIEZA AQUI
2
3
4  #ifndef STACK_H_
5  #define STACK_H_
6
7  typedef struct Node * T_Stack;
8  typedef struct Node {
9    struct Node * next;
10   int number;
11 } T_Node;
12
13 // Creates an empty stack.
14 T_Stack create();
15
16 // Returns true if the stack is empty and false in other case.
17 int isEmpty(T_Stack q);
18
19 // Inserts a number into the stack.
20 void push(T_Stack * pq, int operand);
21
22 // "Inserts" an operator into the stack and operates.
23 // Returns true if everything OK or false in other case.
24 int pushOperator(T_Stack * pq, char operator);
25
26 // Puts into data the number on top of the stack, and removes the top.
27 // Returns true if everything OK or false in other case.
28 int pop(T_Stack * pq, int * data);
29
30 // Frees the memory of a stack and sets it to empty.
31 void destroy(T_Stack * pq);
32
33 #endif /* STACK_H_ */
34
35
36 //TERMINA AQUI
```

```
1  //EMPIEZA AQUI
2
3
4  #ifndef STACK_H_
5  #define STACK_H_
6
7  typedef struct Node * T_Stack;
8  typedef struct Node {
9    struct Node * next;
10   int number;
11 } T_Node;
12
13 // Creates an empty stack.
14 T_Stack create();
15
16 // Returns true if the stack is empty and false in other case.
17 int isEmpty(T_Stack q);
18
19 // Inserts a number into the stack.
20 void push(T_Stack * pq, int operand);
21
22 // "Inserts" an operator into the stack and operates.
23 // Returns true if everything OK or false in other case.
24 int pushOperator(T_Stack * pq, char operator);
25
26 // Puts into data the number on top of the stack, and removes the top.
27 // Returns true if everything OK or false in other case.
28 int pop(T_Stack * pq, int * data);
29
30 // Frees the memory of a stack and sets it to empty.
31 void destroy(T_Stack * pq);
32
33 #endif /* STACK_H_ */
34
35
36 //TERMINA AQUI
```

```
1  //EMPIEZA AQUI
2
3
4  #ifndef STACK_H_
5  #define STACK_H_
6
7  typedef struct Node * T_Stack;
8  typedef struct Node {
9    struct Node * next;
10   int number;
11 } T_Node;
12
13 // Creates an empty stack.
14 T_Stack create();
15
16 // Returns true if the stack is empty and false in other case.
17 int isEmpty(T_Stack q);
18
19 // Inserts a number into the stack.
20 void push(T_Stack * pq, int operand);
21
22 // "Inserts" an operator into the stack and operates.
23 // Returns true if everything OK or false in other case.
24 int pushOperator(T_Stack * pq, char operator);
25
26 // Puts into data the number on top of the stack, and removes the top.
27 // Returns true if everything OK or false in other case.
28 int pop(T_Stack * pq, int * data);
29
30 // Frees the memory of a stack and sets it to empty.
31 void destroy(T_Stack * pq);
32
33 #endif /* STACK_H_ */
34
35
36 //TERMINA AQUI
```

```
1  //EMPIEZA AQUI
2
3
4  #ifndef STACK_H_
5  #define STACK_H_
6
7  typedef struct Node * T_Stack;
8  typedef struct Node {
9    struct Node * next;
10   int number;
11 } T_Node;
12
13 // Creates an empty stack.
14 T_Stack create();
15
16 // Returns true if the stack is empty and false in other case.
17 int isEmpty(T_Stack q);
18
19 // Inserts a number into the stack.
20 void push(T_Stack * pq, int operand);
21
22 // "Inserts" an operator into the stack and operates.
23 // Returns true if everything OK or false in other case.
24 int pushOperator(T_Stack * pq, char operator);
25
26 // Puts into data the number on top of the stack, and removes the top.
27 // Returns true if everything OK or false in other case.
28 int pop(T_Stack * pq, int * data);
29
30 // Frees the memory of a stack and sets it to empty.
31 void destroy(T_Stack * pq);
32
33 #endif /* STACK_H_ */
34
35
36 //TERMINA AQUI
```

```
1  //EMPIEZA AQUI
2
3
4  #ifndef STACK_H_
5  #define STACK_H_
6
7  typedef struct Node * T_Stack;
8  typedef struct Node {
9    struct Node * next;
10   int number;
11 } T_Node;
12
13 // Creates an empty stack.
14 T_Stack create();
15
16 // Returns true if the stack is empty and false in other case.
17 int isEmpty(T_Stack q);
18
19 // Inserts a number into the stack.
20 void push(T_Stack * pq, int operand);
21
22 // "Inserts" an operator into the stack and operates.
23 // Returns true if everything OK or false in other case.
24 int pushOperator(T_Stack * pq, char operator);
25
26 // Puts into data the number on top of the stack, and removes the top.
27 // Returns true if everything OK or false in other case.
28 int pop(T_Stack * pq, int * data);
29
30 // Frees the memory of a stack and sets it to empty.
31 void destroy(T_Stack * pq);
32
33 #endif /* STACK_H_ */
34
35
36 //TERMINA AQUI
```

```
1  //EMPIEZA AQUI
2
3
4  #ifndef STACK_H_
5  #define STACK_H_
6
7  typedef struct Node * T_Stack;
8  typedef struct Node {
9    struct Node * next;
10   int number;
11 } T_Node;
12
13 // Creates an empty stack.
14 T_Stack create();
15
16 // Returns true if the stack is empty and false in other case.
17 int isEmpty(T_Stack q);
18
19 // Inserts a number into the stack.
20 void push(T_Stack * pq, int operand);
21
22 // "Inserts" an operator into the stack and operates.
23 // Returns true if everything OK or false in other case.
24 int pushOperator(T_Stack * pq, char operator);
25
26 // Puts into data the number on top of the stack, and removes the top.
27 // Returns true if everything OK or false in other case.
28 int pop(T_Stack * pq, int * data);
29
30 // Frees the memory of a stack and sets it to empty.
31 void destroy(T_Stack * pq);
32
33 #endif /* STACK_H_ */
34
35
36 //TERMINA AQUI
```

```
1  //EMPIEZA AQUI
2
3
4  #ifndef STACK_H_
5  #define STACK_H_
6
7  typedef struct Node * T_Stack;
8  typedef struct Node {
9    struct Node * next;
10   int number;
11 } T_Node;
12
13 // Creates an empty stack.
14 T_Stack create();
15
16 // Returns true if the stack is empty and false in other case.
17 int isEmpty(T_Stack q);
18
19 // Inserts a number into the stack.
20 void push(T_Stack * pq, int operand);
21
22 // "Inserts" an operator into the stack and operates.
23 // Returns true if everything OK or false in other case.
24 int pushOperator(T_Stack * pq, char operator);
25
26 // Puts into data the number on top of the stack, and removes the top.
27 // Returns true if everything OK or false in other case.
28 int pop(T_Stack * pq, int * data);
29
30 // Frees the memory of a stack and sets it to empty.
31 void destroy(T_Stack * pq);
32
33 #endif /* STACK_H_ */
34
35
36 //TERMINA AQUI
```

```
1  //EMPIEZA AQUI
2
3
4  #ifndef STACK_H_
5  #define STACK_H_
6
7  typedef struct Node * T_Stack;
8  typedef struct Node {
9    struct Node * next;
10   int number;
11 } T_Node;
12
13 // Creates an empty stack.
14 T_Stack create();
15
16 // Returns true if the stack is empty and false in other case.
17 int isEmpty(T_Stack q);
18
19 // Inserts a number into the stack.
20 void push(T_Stack * pq, int operand);
21
22 // "Inserts" an operator into the stack and operates.
23 // Returns true if everything OK or false in other case.
24 int pushOperator(T_Stack * pq, char operator);
25
26 // Puts into data the number on top of the stack, and removes the top.
27 // Returns true if everything OK or false in other case.
28 int pop(T_Stack * pq, int * data);
29
30 // Frees the memory of a stack and sets it to empty.
31 void destroy(T_Stack * pq);
32
33 #endif /* STACK_H_ */
34
35
36 //TERMINA AQUI
```

```
1  //EMPIEZA AQUI
2
3
4  #ifndef STACK_H_
5  #define STACK_H_
6
7  typedef struct Node * T_Stack;
8  typedef struct Node {
9    struct Node * next;
10   int number;
11 } T_Node;
12
13 // Creates an empty stack.
14 T_Stack create();
15
16 // Returns true if the stack is empty and false in other case.
17 int isEmpty(T_Stack q);
18
19 // Inserts a number into the stack.
20 void push(T_Stack * pq, int operand);
21
22 // "Inserts" an operator into the stack and operates.
23 // Returns true if everything OK or false in other case.
24 int pushOperator(T_Stack * pq, char operator);
25
26 // Puts into data the number on top of the stack, and removes the top.
27 // Returns true if everything OK or false in other case.
28 int pop(T_Stack * pq, int * data);
29
30 // Frees the memory of a stack and sets it to empty.
31 void destroy(T_Stack * pq);
32
33 #endif /* STACK_H_ */
34
35
36 //TERMINA AQUI
```

```
1  //EMPIEZA AQUI
2
3
4  #ifndef STACK_H_
5  #define STACK_H_
6
7  typedef struct Node * T_Stack;
8  typedef struct Node {
9    struct Node * next;
10   int number;
11 } T_Node;
12
13 // Creates an empty stack.
14 T_Stack create();
15
16 // Returns true if the stack is empty and false in other case.
17 int isEmpty(T_Stack q);
18
19 // Inserts a number into the stack.
20 void push(T_Stack * pq, int operand);
21
22 // "Inserts" an operator into the stack and operates.
23 // Returns true if everything OK or false in other case.
24 int pushOperator(T_Stack * pq, char operator);
25
26 // Puts into data the number on top of the stack, and removes the top.
27 // Returns true if everything OK or false in other case.
28 int pop(T_Stack * pq, int * data);
29
30 // Frees the memory of a stack and sets it to empty.
31 void destroy(T_Stack * pq);
32
33 #endif /* STACK_H_ */
34
35
36 //TERMINA AQUI
```

```
1  //EMPIEZA AQUI
2
3
4  #ifndef STACK_H_
5  #define STACK_H_
6
7  typedef struct Node * T_Stack;
8  typedef struct Node {
9    struct Node * next;
10   int number;
11 } T_Node;
12
13 // Creates an empty stack.
14 T_Stack create();
15
16 // Returns true if the stack is empty and false in other case.
17 int isEmpty(T_Stack q);
18
19 // Inserts a number into the stack.
20 void push(T_Stack * pq, int operand);
21
22 // "Inserts" an operator into the stack and operates.
23 // Returns true if everything OK or false in other case.
24 int pushOperator(T_Stack * pq, char operator);
25
26 // Puts into data the number on top of the stack, and removes the top.
27 // Returns true if everything OK or false in other case.
28 int pop(T_Stack * pq, int * data);
29
30 // Frees the memory of a stack and sets it to empty.
31 void destroy(T_Stack * pq);
32
33 #endif /* STACK_H_ */
34
35
36 //TERMINA AQUI
```

```
1  //EMPIEZA AQUI
2
3
4  #ifndef STACK_H_
5  #define STACK_H_
6
7  typedef struct Node * T_Stack;
8  typedef struct Node {
9    struct Node * next;
10   int number;
11 } T_Node;
12
13 // Creates an empty stack.
14 T_Stack create();
15
16 // Returns true if the stack is empty and false in other case.
17 int isEmpty(T_Stack q);
18
19 // Inserts a number into the stack.
20 void push(T_Stack * pq, int operand);
21
22 // "Inserts" an operator into the stack and operates.
23 // Returns true if everything OK or false in other case.
24 int pushOperator(T_Stack * pq, char operator);
25
26 // Puts into data the number on top of the stack, and removes the top.
27 // Returns true if everything OK or false in other case.
28 int pop(T_Stack * pq, int * data);
29
30 // Frees the memory of a stack and sets it to empty.
31 void destroy(T_Stack * pq);
32
33 #endif /* STACK_H_ */
34
35
36 //TERMINA AQUI
```