# STACKC.pdf


**Ferre18**


**Programación de Sistemas y Concurrencia**

**2º Grado en Ingeniería Informática**

**Escuela Técnica Superior de Ingeniería Informática**
**Universidad de Málaga**

```c
//EMPIEZA AQUI


#include <stdio.h>
#include <stdlib.h>
#include "Stack.h"

// Creates an empty stack.
T_Stack create() {
  return NULL;
}

// Returns true if the stack is empty and false in other case.
int isEmpty(T_Stack q) {
  return q == NULL ? 1 : 0;
}

// Inserts a number into the stack.
void push(T_Stack *pq, int operand) {
  T_Stack nuevo = (T_Stack) malloc(sizeof(struct Node));
  if (nuevo == NULL) {
    perror("Error al insertar");
  } else {
    nuevo->number = operand;

    if (*pq == NULL) {
      nuevo->next = NULL;
    } else {
      nuevo->next = *pq;
    }
    *pq = nuevo;
  }
}

// "Inserts" an operator into the stack and operates.
// Returns true if everything OK or false in other case.
int pushOperator(T_Stack *pq, char operator) {
  int count2 = 0;
  T_Stack iter = *pq;
  while (iter != NULL && count2 < 2) {
    iter = iter->next;
    count2++;
  }

  if (count2 != 2) {
    return 0;
  } else {
    int fst, snd;
    pop(pq, &snd);
    pop(pq, &fst);
    if (operator == '+') {
      push(pq, fst + snd);
    } else if (operator == '-') {
      push(pq, fst - snd);
    } else if (operator == '*') {
      push(pq, fst * snd);
    } else if (operator == '/') {
      push(pq, fst / snd);
    } else {
```

```
60        return 0;
61      }
62      return 1;
63    }
64  }
65
66  // Puts into data the number on top of the stack, and removes the top.
67  // Returns true if everything OK or false in other case.
68  int pop(T_Stack *pq, int *data) {
69    if (*pq == NULL) {
70      return 0;
71    } else {
72      *data = (*pq)->number;
73      *pq = (*pq)->next;
74      return 1;
75    }
76  }
77
78  // Frees the memory of a stack and sets it to empty.
79  void destroy(T_Stack *pq) {
80    while(*pq != NULL) {
81      T_Stack delete = *pq;
82      *pq = (*pq)->next;
83      free(delete);
84    }
85  }
86
87
88  //TERMINA AQUI
```

```c
//EMPIEZA AQUI


#include <stdio.h>
#include <stdlib.h>
#include "Stack.h"

// Creates an empty stack.
T_Stack create() {
  return NULL;
}

// Returns true if the stack is empty and false in other case.
int isEmpty(T_Stack q) {
  return q == NULL ? 1 : 0;
}

// Inserts a number into the stack.
void push(T_Stack *pq, int operand) {
  T_Stack nuevo = (T_Stack) malloc(sizeof(struct Node));
  if (nuevo == NULL) {
    perror("Error al insertar");
  } else {
    nuevo->number = operand;

    if (*pq == NULL) {
      nuevo->next = NULL;
    } else {
      nuevo->next = *pq;
    }
    *pq = nuevo;
  }
}

// "Inserts" an operator into the stack and operates.
// Returns true if everything OK or false in other case.
int pushOperator(T_Stack *pq, char operator) {
  int count2 = 0;
  T_Stack iter = *pq;
  while (iter != NULL && count2 < 2) {
    iter = iter->next;
    count2++;
  }

  if (count2 != 2) {
    return 0;
  } else {
    int fst, snd;
    pop(pq, &snd);
    pop(pq, &fst);
    if (operator == '+') {
      push(pq, fst + snd);
    } else if (operator == '-') {
      push(pq, fst - snd);
    } else if (operator == '*') {
      push(pq, fst * snd);
    } else if (operator == '/') {
      push(pq, fst / snd);
    } else {
```

```
60        return 0;
61      }
62      return 1;
63    }
64  }
65
66  // Puts into data the number on top of the stack, and removes the top.
67  // Returns true if everything OK or false in other case.
68  int pop(T_Stack *pq, int *data) {
69    if (*pq == NULL) {
70      return 0;
71    } else {
72      *data = (*pq)->number;
73      *pq = (*pq)->next;
74      return 1;
75    }
76  }
77
78  // Frees the memory of a stack and sets it to empty.
79  void destroy(T_Stack *pq) {
80    while(*pq != NULL) {
81      T_Stack delete = *pq;
82      *pq = (*pq)->next;
83      free(delete);
84    }
85  }
86
87
88  //TERMINA AQUI
```

```c
//EMPIEZA AQUI


#include <stdio.h>
#include <stdlib.h>
#include "Stack.h"

// Creates an empty stack.
T_Stack create() {
  return NULL;
}

// Returns true if the stack is empty and false in other case.
int isEmpty(T_Stack q) {
  return q == NULL ? 1 : 0;
}

// Inserts a number into the stack.
void push(T_Stack *pq, int operand) {
  T_Stack nuevo = (T_Stack) malloc(sizeof(struct Node));
  if (nuevo == NULL) {
    perror("Error al insertar");
  } else {
    nuevo->number = operand;

    if (*pq == NULL) {
      nuevo->next = NULL;
    } else {
      nuevo->next = *pq;
    }
    *pq = nuevo;
  }
}

// "Inserts" an operator into the stack and operates.
// Returns true if everything OK or false in other case.
int pushOperator(T_Stack *pq, char operator) {
  int count2 = 0;
  T_Stack iter = *pq;
  while (iter != NULL && count2 < 2) {
    iter = iter->next;
    count2++;
  }

  if (count2 != 2) {
    return 0;
  } else {
    int fst, snd;
    pop(pq, &snd);
    pop(pq, &fst);
    if (operator == '+') {
      push(pq, fst + snd);
    } else if (operator == '-') {
      push(pq, fst - snd);
    } else if (operator == '*') {
      push(pq, fst * snd);
    } else if (operator == '/') {
      push(pq, fst / snd);
    } else {
```

```
60        return 0;
61      }
62      return 1;
63    }
64  }
65
66  // Puts into data the number on top of the stack, and removes the top.
67  // Returns true if everything OK or false in other case.
68  int pop(T_Stack *pq, int *data) {
69    if (*pq == NULL) {
70      return 0;
71    } else {
72      *data = (*pq)->number;
73      *pq = (*pq)->next;
74      return 1;
75    }
76  }
77
78  // Frees the memory of a stack and sets it to empty.
79  void destroy(T_Stack *pq) {
80    while(*pq != NULL) {
81      T_Stack delete = *pq;
82      *pq = (*pq)->next;
83      free(delete);
84    }
85  }
86
87
88  //TERMINA AQUI
```

```c
//EMPIEZA AQUI


#include <stdio.h>
#include <stdlib.h>
#include "Stack.h"

// Creates an empty stack.
T_Stack create() {
  return NULL;
}

// Returns true if the stack is empty and false in other case.
int isEmpty(T_Stack q) {
  return q == NULL ? 1 : 0;
}

// Inserts a number into the stack.
void push(T_Stack *pq, int operand) {
  T_Stack nuevo = (T_Stack) malloc(sizeof(struct Node));
  if (nuevo == NULL) {
    perror("Error al insertar");
  } else {
    nuevo->number = operand;

    if (*pq == NULL) {
      nuevo->next = NULL;
    } else {
      nuevo->next = *pq;
    }
    *pq = nuevo;
  }
}

// "Inserts" an operator into the stack and operates.
// Returns true if everything OK or false in other case.
int pushOperator(T_Stack *pq, char operator) {
  int count2 = 0;
  T_Stack iter = *pq;
  while (iter != NULL && count2 < 2) {
    iter = iter->next;
    count2++;
  }

  if (count2 != 2) {
    return 0;
  } else {
    int fst, snd;
    pop(pq, &snd);
    pop(pq, &fst);
    if (operator == '+') {
      push(pq, fst + snd);
    } else if (operator == '-') {
      push(pq, fst - snd);
    } else if (operator == '*') {
      push(pq, fst * snd);
    } else if (operator == '/') {
      push(pq, fst / snd);
    } else {
```

```
60        return 0;
61      }
62      return 1;
63    }
64  }
65
66  // Puts into data the number on top of the stack, and removes the top.
67  // Returns true if everything OK or false in other case.
68  int pop(T_Stack *pq, int *data) {
69    if (*pq == NULL) {
70      return 0;
71    } else {
72      *data = (*pq)->number;
73      *pq = (*pq)->next;
74      return 1;
75    }
76  }
77
78  // Frees the memory of a stack and sets it to empty.
79  void destroy(T_Stack *pq) {
80    while(*pq != NULL) {
81      T_Stack delete = *pq;
82      *pq = (*pq)->next;
83      free(delete);
84    }
85  }
86
87
88  //TERMINA AQUI
```

```c
//EMPIEZA AQUI


#include <stdio.h>
#include <stdlib.h>
#include "Stack.h"

// Creates an empty stack.
T_Stack create() {
  return NULL;
}

// Returns true if the stack is empty and false in other case.
int isEmpty(T_Stack q) {
  return q == NULL ? 1 : 0;
}

// Inserts a number into the stack.
void push(T_Stack *pq, int operand) {
  T_Stack nuevo = (T_Stack) malloc(sizeof(struct Node));
  if (nuevo == NULL) {
    perror("Error al insertar");
  } else {
    nuevo->number = operand;

    if (*pq == NULL) {
      nuevo->next = NULL;
    } else {
      nuevo->next = *pq;
    }
    *pq = nuevo;
  }
}

// "Inserts" an operator into the stack and operates.
// Returns true if everything OK or false in other case.
int pushOperator(T_Stack *pq, char operator) {
  int count2 = 0;
  T_Stack iter = *pq;
  while (iter != NULL && count2 < 2) {
    iter = iter->next;
    count2++;
  }

  if (count2 != 2) {
    return 0;
  } else {
    int fst, snd;
    pop(pq, &snd);
    pop(pq, &fst);
    if (operator == '+') {
      push(pq, fst + snd);
    } else if (operator == '-') {
      push(pq, fst - snd);
    } else if (operator == '*') {
      push(pq, fst * snd);
    } else if (operator == '/') {
      push(pq, fst / snd);
    } else {
```

```
60        return 0;
61      }
62      return 1;
63    }
64  }
65
66  // Puts into data the number on top of the stack, and removes the top.
67  // Returns true if everything OK or false in other case.
68  int pop(T_Stack *pq, int *data) {
69    if (*pq == NULL) {
70      return 0;
71    } else {
72      *data = (*pq)->number;
73      *pq = (*pq)->next;
74      return 1;
75    }
76  }
77
78  // Frees the memory of a stack and sets it to empty.
79  void destroy(T_Stack *pq) {
80    while(*pq != NULL) {
81      T_Stack delete = *pq;
82      *pq = (*pq)->next;
83      free(delete);
84    }
85  }
86
87
88  //TERMINA AQUI
```

```c
//EMPIEZA AQUI


#include <stdio.h>
#include <stdlib.h>
#include "Stack.h"

// Creates an empty stack.
T_Stack create() {
  return NULL;
}

// Returns true if the stack is empty and false in other case.
int isEmpty(T_Stack q) {
  return q == NULL ? 1 : 0;
}

// Inserts a number into the stack.
void push(T_Stack *pq, int operand) {
  T_Stack nuevo = (T_Stack) malloc(sizeof(struct Node));
  if (nuevo == NULL) {
    perror("Error al insertar");
  } else {
    nuevo->number = operand;

    if (*pq == NULL) {
      nuevo->next = NULL;
    } else {
      nuevo->next = *pq;
    }
    *pq = nuevo;
  }
}

// "Inserts" an operator into the stack and operates.
// Returns true if everything OK or false in other case.
int pushOperator(T_Stack *pq, char operator) {
  int count2 = 0;
  T_Stack iter = *pq;
  while (iter != NULL && count2 < 2) {
    iter = iter->next;
    count2++;
  }

  if (count2 != 2) {
    return 0;
  } else {
    int fst, snd;
    pop(pq, &snd);
    pop(pq, &fst);
    if (operator == '+') {
      push(pq, fst + snd);
    } else if (operator == '-') {
      push(pq, fst - snd);
    } else if (operator == '*') {
      push(pq, fst * snd);
    } else if (operator == '/') {
      push(pq, fst / snd);
    } else {
```

```
60        return 0;
61      }
62      return 1;
63    }
64  }
65
66  // Puts into data the number on top of the stack, and removes the top.
67  // Returns true if everything OK or false in other case.
68  int pop(T_Stack *pq, int *data) {
69    if (*pq == NULL) {
70      return 0;
71    } else {
72      *data = (*pq)->number;
73      *pq = (*pq)->next;
74      return 1;
75    }
76  }
77
78  // Frees the memory of a stack and sets it to empty.
79  void destroy(T_Stack *pq) {
80    while(*pq != NULL) {
81      T_Stack delete = *pq;
82      *pq = (*pq)->next;
83      free(delete);
84    }
85  }
86
87
88  //TERMINA AQUI
```

```c
//EMPIEZA AQUI


#include <stdio.h>
#include <stdlib.h>
#include "Stack.h"

// Creates an empty stack.
T_Stack create() {
  return NULL;
}

// Returns true if the stack is empty and false in other case.
int isEmpty(T_Stack q) {
  return q == NULL ? 1 : 0;
}

// Inserts a number into the stack.
void push(T_Stack *pq, int operand) {
  T_Stack nuevo = (T_Stack) malloc(sizeof(struct Node));
  if (nuevo == NULL) {
    perror("Error al insertar");
  } else {
    nuevo->number = operand;

    if (*pq == NULL) {
      nuevo->next = NULL;
    } else {
      nuevo->next = *pq;
    }
    *pq = nuevo;
  }
}

// "Inserts" an operator into the stack and operates.
// Returns true if everything OK or false in other case.
int pushOperator(T_Stack *pq, char operator) {
  int count2 = 0;
  T_Stack iter = *pq;
  while (iter != NULL && count2 < 2) {
    iter = iter->next;
    count2++;
  }

  if (count2 != 2) {
    return 0;
  } else {
    int fst, snd;
    pop(pq, &snd);
    pop(pq, &fst);
    if (operator == '+') {
      push(pq, fst + snd);
    } else if (operator == '-') {
      push(pq, fst - snd);
    } else if (operator == '*') {
      push(pq, fst * snd);
    } else if (operator == '/') {
      push(pq, fst / snd);
    } else {
```

```
60        return 0;
61      }
62      return 1;
63    }
64  }
65
66  // Puts into data the number on top of the stack, and removes the top.
67  // Returns true if everything OK or false in other case.
68  int pop(T_Stack *pq, int *data) {
69    if (*pq == NULL) {
70      return 0;
71    } else {
72      *data = (*pq)->number;
73      *pq = (*pq)->next;
74      return 1;
75    }
76  }
77
78  // Frees the memory of a stack and sets it to empty.
79  void destroy(T_Stack *pq) {
80    while(*pq != NULL) {
81      T_Stack delete = *pq;
82      *pq = (*pq)->next;
83      free(delete);
84    }
85  }
86
87
88  //TERMINA AQUI
```

```c
//EMPIEZA AQUI


#include <stdio.h>
#include <stdlib.h>
#include "Stack.h"

// Creates an empty stack.
T_Stack create() {
  return NULL;
}

// Returns true if the stack is empty and false in other case.
int isEmpty(T_Stack q) {
  return q == NULL ? 1 : 0;
}

// Inserts a number into the stack.
void push(T_Stack *pq, int operand) {
  T_Stack nuevo = (T_Stack) malloc(sizeof(struct Node));
  if (nuevo == NULL) {
    perror("Error al insertar");
  } else {
    nuevo->number = operand;

    if (*pq == NULL) {
      nuevo->next = NULL;
    } else {
      nuevo->next = *pq;
    }
    *pq = nuevo;
  }
}

// "Inserts" an operator into the stack and operates.
// Returns true if everything OK or false in other case.
int pushOperator(T_Stack *pq, char operator) {
  int count2 = 0;
  T_Stack iter = *pq;
  while (iter != NULL && count2 < 2) {
    iter = iter->next;
    count2++;
  }

  if (count2 != 2) {
    return 0;
  } else {
    int fst, snd;
    pop(pq, &snd);
    pop(pq, &fst);
    if (operator == '+') {
      push(pq, fst + snd);
    } else if (operator == '-') {
      push(pq, fst - snd);
    } else if (operator == '*') {
      push(pq, fst * snd);
    } else if (operator == '/') {
      push(pq, fst / snd);
    } else {
```

```
60        return 0;
61      }
62      return 1;
63    }
64  }
65
66  // Puts into data the number on top of the stack, and removes the top.
67  // Returns true if everything OK or false in other case.
68  int pop(T_Stack *pq, int *data) {
69    if (*pq == NULL) {
70      return 0;
71    } else {
72      *data = (*pq)->number;
73      *pq = (*pq)->next;
74      return 1;
75    }
76  }
77
78  // Frees the memory of a stack and sets it to empty.
79  void destroy(T_Stack *pq) {
80    while(*pq != NULL) {
81      T_Stack delete = *pq;
82      *pq = (*pq)->next;
83      free(delete);
84    }
85  }
86
87
88  //TERMINA AQUI
```

```c
//EMPIEZA AQUI


#include <stdio.h>
#include <stdlib.h>
#include "Stack.h"

// Creates an empty stack.
T_Stack create() {
  return NULL;
}

// Returns true if the stack is empty and false in other case.
int isEmpty(T_Stack q) {
  return q == NULL ? 1 : 0;
}

// Inserts a number into the stack.
void push(T_Stack *pq, int operand) {
  T_Stack nuevo = (T_Stack) malloc(sizeof(struct Node));
  if (nuevo == NULL) {
    perror("Error al insertar");
  } else {
    nuevo->number = operand;

    if (*pq == NULL) {
      nuevo->next = NULL;
    } else {
      nuevo->next = *pq;
    }
    *pq = nuevo;
  }
}

// "Inserts" an operator into the stack and operates.
// Returns true if everything OK or false in other case.
int pushOperator(T_Stack *pq, char operator) {
  int count2 = 0;
  T_Stack iter = *pq;
  while (iter != NULL && count2 < 2) {
    iter = iter->next;
    count2++;
  }

  if (count2 != 2) {
    return 0;
  } else {
    int fst, snd;
    pop(pq, &snd);
    pop(pq, &fst);
    if (operator == '+') {
      push(pq, fst + snd);
    } else if (operator == '-') {
      push(pq, fst - snd);
    } else if (operator == '*') {
      push(pq, fst * snd);
    } else if (operator == '/') {
      push(pq, fst / snd);
    } else {
```

```
60        return 0;
61      }
62      return 1;
63    }
64  }
65
66  // Puts into data the number on top of the stack, and removes the top.
67  // Returns true if everything OK or false in other case.
68  int pop(T_Stack *pq, int *data) {
69    if (*pq == NULL) {
70      return 0;
71    } else {
72      *data = (*pq)->number;
73      *pq = (*pq)->next;
74      return 1;
75    }
76  }
77
78  // Frees the memory of a stack and sets it to empty.
79  void destroy(T_Stack *pq) {
80    while(*pq != NULL) {
81      T_Stack delete = *pq;
82      *pq = (*pq)->next;
83      free(delete);
84    }
85  }
86
87
88  //TERMINA AQUI
```

```c
//EMPIEZA AQUI


#include <stdio.h>
#include <stdlib.h>
#include "Stack.h"

// Creates an empty stack.
T_Stack create() {
  return NULL;
}

// Returns true if the stack is empty and false in other case.
int isEmpty(T_Stack q) {
  return q == NULL ? 1 : 0;
}

// Inserts a number into the stack.
void push(T_Stack *pq, int operand) {
  T_Stack nuevo = (T_Stack) malloc(sizeof(struct Node));
  if (nuevo == NULL) {
    perror("Error al insertar");
  } else {
    nuevo->number = operand;

    if (*pq == NULL) {
      nuevo->next = NULL;
    } else {
      nuevo->next = *pq;
    }
    *pq = nuevo;
  }
}

// "Inserts" an operator into the stack and operates.
// Returns true if everything OK or false in other case.
int pushOperator(T_Stack *pq, char operator) {
  int count2 = 0;
  T_Stack iter = *pq;
  while (iter != NULL && count2 < 2) {
    iter = iter->next;
    count2++;
  }

  if (count2 != 2) {
    return 0;
  } else {
    int fst, snd;
    pop(pq, &snd);
    pop(pq, &fst);
    if (operator == '+') {
      push(pq, fst + snd);
    } else if (operator == '-') {
      push(pq, fst - snd);
    } else if (operator == '*') {
      push(pq, fst * snd);
    } else if (operator == '/') {
      push(pq, fst / snd);
    } else {
```

```
60        return 0;
61      }
62      return 1;
63    }
64  }
65
66  // Puts into data the number on top of the stack, and removes the top.
67  // Returns true if everything OK or false in other case.
68  int pop(T_Stack *pq, int *data) {
69    if (*pq == NULL) {
70      return 0;
71    } else {
72      *data = (*pq)->number;
73      *pq = (*pq)->next;
74      return 1;
75    }
76  }
77
78  // Frees the memory of a stack and sets it to empty.
79  void destroy(T_Stack *pq) {
80    while(*pq != NULL) {
81      T_Stack delete = *pq;
82      *pq = (*pq)->next;
83      free(delete);
84    }
85  }
86
87
88  //TERMINA AQUI
```

```c
//EMPIEZA AQUI


#include <stdio.h>
#include <stdlib.h>
#include "Stack.h"

// Creates an empty stack.
T_Stack create() {
  return NULL;
}

// Returns true if the stack is empty and false in other case.
int isEmpty(T_Stack q) {
  return q == NULL ? 1 : 0;
}

// Inserts a number into the stack.
void push(T_Stack *pq, int operand) {
  T_Stack nuevo = (T_Stack) malloc(sizeof(struct Node));
  if (nuevo == NULL) {
    perror("Error al insertar");
  } else {
    nuevo->number = operand;

    if (*pq == NULL) {
      nuevo->next = NULL;
    } else {
      nuevo->next = *pq;
    }
    *pq = nuevo;
  }
}

// "Inserts" an operator into the stack and operates.
// Returns true if everything OK or false in other case.
int pushOperator(T_Stack *pq, char operator) {
  int count2 = 0;
  T_Stack iter = *pq;
  while (iter != NULL && count2 < 2) {
    iter = iter->next;
    count2++;
  }

  if (count2 != 2) {
    return 0;
  } else {
    int fst, snd;
    pop(pq, &snd);
    pop(pq, &fst);
    if (operator == '+') {
      push(pq, fst + snd);
    } else if (operator == '-') {
      push(pq, fst - snd);
    } else if (operator == '*') {
      push(pq, fst * snd);
    } else if (operator == '/') {
      push(pq, fst / snd);
    } else {
```

```
60        return 0;
61      }
62      return 1;
63    }
64  }
65
66  // Puts into data the number on top of the stack, and removes the top.
67  // Returns true if everything OK or false in other case.
68  int pop(T_Stack *pq, int *data) {
69    if (*pq == NULL) {
70      return 0;
71    } else {
72      *data = (*pq)->number;
73      *pq = (*pq)->next;
74      return 1;
75    }
76  }
77
78  // Frees the memory of a stack and sets it to empty.
79  void destroy(T_Stack *pq) {
80    while(*pq != NULL) {
81      T_Stack delete = *pq;
82      *pq = (*pq)->next;
83      free(delete);
84    }
85  }
86
87
88  //TERMINA AQUI
```

```c
//EMPIEZA AQUI


#include <stdio.h>
#include <stdlib.h>
#include "Stack.h"

// Creates an empty stack.
T_Stack create() {
  return NULL;
}

// Returns true if the stack is empty and false in other case.
int isEmpty(T_Stack q) {
  return q == NULL ? 1 : 0;
}

// Inserts a number into the stack.
void push(T_Stack *pq, int operand) {
  T_Stack nuevo = (T_Stack) malloc(sizeof(struct Node));
  if (nuevo == NULL) {
    perror("Error al insertar");
  } else {
    nuevo->number = operand;

    if (*pq == NULL) {
      nuevo->next = NULL;
    } else {
      nuevo->next = *pq;
    }
    *pq = nuevo;
  }
}

// "Inserts" an operator into the stack and operates.
// Returns true if everything OK or false in other case.
int pushOperator(T_Stack *pq, char operator) {
  int count2 = 0;
  T_Stack iter = *pq;
  while (iter != NULL && count2 < 2) {
    iter = iter->next;
    count2++;
  }

  if (count2 != 2) {
    return 0;
  } else {
    int fst, snd;
    pop(pq, &snd);
    pop(pq, &fst);
    if (operator == '+') {
      push(pq, fst + snd);
    } else if (operator == '-') {
      push(pq, fst - snd);
    } else if (operator == '*') {
      push(pq, fst * snd);
    } else if (operator == '/') {
      push(pq, fst / snd);
    } else {
```

```
60        return 0;
61      }
62      return 1;
63    }
64  }
65
66  // Puts into data the number on top of the stack, and removes the top.
67  // Returns true if everything OK or false in other case.
68  int pop(T_Stack *pq, int *data) {
69    if (*pq == NULL) {
70      return 0;
71    } else {
72      *data = (*pq)->number;
73      *pq = (*pq)->next;
74      return 1;
75    }
76  }
77
78  // Frees the memory of a stack and sets it to empty.
79  void destroy(T_Stack *pq) {
80    while(*pq != NULL) {
81      T_Stack delete = *pq;
82      *pq = (*pq)->next;
83      free(delete);
84    }
85  }
86
87
88  //TERMINA AQUI
```

```c
//EMPIEZA AQUI


#include <stdio.h>
#include <stdlib.h>
#include "Stack.h"

// Creates an empty stack.
T_Stack create() {
  return NULL;
}

// Returns true if the stack is empty and false in other case.
int isEmpty(T_Stack q) {
  return q == NULL ? 1 : 0;
}

// Inserts a number into the stack.
void push(T_Stack *pq, int operand) {
  T_Stack nuevo = (T_Stack) malloc(sizeof(struct Node));
  if (nuevo == NULL) {
    perror("Error al insertar");
  } else {
    nuevo->number = operand;

    if (*pq == NULL) {
      nuevo->next = NULL;
    } else {
      nuevo->next = *pq;
    }
    *pq = nuevo;
  }
}

// "Inserts" an operator into the stack and operates.
// Returns true if everything OK or false in other case.
int pushOperator(T_Stack *pq, char operator) {
  int count2 = 0;
  T_Stack iter = *pq;
  while (iter != NULL && count2 < 2) {
    iter = iter->next;
    count2++;
  }

  if (count2 != 2) {
    return 0;
  } else {
    int fst, snd;
    pop(pq, &snd);
    pop(pq, &fst);
    if (operator == '+') {
      push(pq, fst + snd);
    } else if (operator == '-') {
      push(pq, fst - snd);
    } else if (operator == '*') {
      push(pq, fst * snd);
    } else if (operator == '/') {
      push(pq, fst / snd);
    } else {
```

```
60        return 0;
61      }
62      return 1;
63    }
64  }
65
66  // Puts into data the number on top of the stack, and removes the top.
67  // Returns true if everything OK or false in other case.
68  int pop(T_Stack *pq, int *data) {
69    if (*pq == NULL) {
70      return 0;
71    } else {
72      *data = (*pq)->number;
73      *pq = (*pq)->next;
74      return 1;
75    }
76  }
77
78  // Frees the memory of a stack and sets it to empty.
79  void destroy(T_Stack *pq) {
80    while(*pq != NULL) {
81      T_Stack delete = *pq;
82      *pq = (*pq)->next;
83      free(delete);
84    }
85  }
86
87
88  //TERMINA AQUI
```

```c
//EMPIEZA AQUI


#include <stdio.h>
#include <stdlib.h>
#include "Stack.h"

// Creates an empty stack.
T_Stack create() {
  return NULL;
}

// Returns true if the stack is empty and false in other case.
int isEmpty(T_Stack q) {
  return q == NULL ? 1 : 0;
}

// Inserts a number into the stack.
void push(T_Stack *pq, int operand) {
  T_Stack nuevo = (T_Stack) malloc(sizeof(struct Node));
  if (nuevo == NULL) {
    perror("Error al insertar");
  } else {
    nuevo->number = operand;

    if (*pq == NULL) {
      nuevo->next = NULL;
    } else {
      nuevo->next = *pq;
    }
    *pq = nuevo;
  }
}

// "Inserts" an operator into the stack and operates.
// Returns true if everything OK or false in other case.
int pushOperator(T_Stack *pq, char operator) {
  int count2 = 0;
  T_Stack iter = *pq;
  while (iter != NULL && count2 < 2) {
    iter = iter->next;
    count2++;
  }

  if (count2 != 2) {
    return 0;
  } else {
    int fst, snd;
    pop(pq, &snd);
    pop(pq, &fst);
    if (operator == '+') {
      push(pq, fst + snd);
    } else if (operator == '-') {
      push(pq, fst - snd);
    } else if (operator == '*') {
      push(pq, fst * snd);
    } else if (operator == '/') {
      push(pq, fst / snd);
    } else {
```

```
60        return 0;
61      }
62      return 1;
63    }
64  }
65
66  // Puts into data the number on top of the stack, and removes the top.
67  // Returns true if everything OK or false in other case.
68  int pop(T_Stack *pq, int *data) {
69    if (*pq == NULL) {
70      return 0;
71    } else {
72      *data = (*pq)->number;
73      *pq = (*pq)->next;
74      return 1;
75    }
76  }
77
78  // Frees the memory of a stack and sets it to empty.
79  void destroy(T_Stack *pq) {
80    while(*pq != NULL) {
81      T_Stack delete = *pq;
82      *pq = (*pq)->next;
83      free(delete);
84    }
85  }
86
87
88  //TERMINA AQUI
```

```c
//EMPIEZA AQUI


#include <stdio.h>
#include <stdlib.h>
#include "Stack.h"

// Creates an empty stack.
T_Stack create() {
  return NULL;
}

// Returns true if the stack is empty and false in other case.
int isEmpty(T_Stack q) {
  return q == NULL ? 1 : 0;
}

// Inserts a number into the stack.
void push(T_Stack *pq, int operand) {
  T_Stack nuevo = (T_Stack) malloc(sizeof(struct Node));
  if (nuevo == NULL) {
    perror("Error al insertar");
  } else {
    nuevo->number = operand;

    if (*pq == NULL) {
      nuevo->next = NULL;
    } else {
      nuevo->next = *pq;
    }
    *pq = nuevo;
  }
}

// "Inserts" an operator into the stack and operates.
// Returns true if everything OK or false in other case.
int pushOperator(T_Stack *pq, char operator) {
  int count2 = 0;
  T_Stack iter = *pq;
  while (iter != NULL && count2 < 2) {
    iter = iter->next;
    count2++;
  }

  if (count2 != 2) {
    return 0;
  } else {
    int fst, snd;
    pop(pq, &snd);
    pop(pq, &fst);
    if (operator == '+') {
      push(pq, fst + snd);
    } else if (operator == '-') {
      push(pq, fst - snd);
    } else if (operator == '*') {
      push(pq, fst * snd);
    } else if (operator == '/') {
      push(pq, fst / snd);
    } else {
```

```
60        return 0;
61      }
62      return 1;
63    }
64  }
65
66  // Puts into data the number on top of the stack, and removes the top.
67  // Returns true if everything OK or false in other case.
68  int pop(T_Stack *pq, int *data) {
69    if (*pq == NULL) {
70      return 0;
71    } else {
72      *data = (*pq)->number;
73      *pq = (*pq)->next;
74      return 1;
75    }
76  }
77
78  // Frees the memory of a stack and sets it to empty.
79  void destroy(T_Stack *pq) {
80    while(*pq != NULL) {
81      T_Stack delete = *pq;
82      *pq = (*pq)->next;
83      free(delete);
84    }
85  }
86
87
88  //TERMINA AQUI
```

```c
//EMPIEZA AQUI


#include <stdio.h>
#include <stdlib.h>
#include "Stack.h"

// Creates an empty stack.
T_Stack create() {
  return NULL;
}

// Returns true if the stack is empty and false in other case.
int isEmpty(T_Stack q) {
  return q == NULL ? 1 : 0;
}

// Inserts a number into the stack.
void push(T_Stack *pq, int operand) {
  T_Stack nuevo = (T_Stack) malloc(sizeof(struct Node));
  if (nuevo == NULL) {
    perror("Error al insertar");
  } else {
    nuevo->number = operand;

    if (*pq == NULL) {
      nuevo->next = NULL;
    } else {
      nuevo->next = *pq;
    }
    *pq = nuevo;
  }
}

// "Inserts" an operator into the stack and operates.
// Returns true if everything OK or false in other case.
int pushOperator(T_Stack *pq, char operator) {
  int count2 = 0;
  T_Stack iter = *pq;
  while (iter != NULL && count2 < 2) {
    iter = iter->next;
    count2++;
  }

  if (count2 != 2) {
    return 0;
  } else {
    int fst, snd;
    pop(pq, &snd);
    pop(pq, &fst);
    if (operator == '+') {
      push(pq, fst + snd);
    } else if (operator == '-') {
      push(pq, fst - snd);
    } else if (operator == '*') {
      push(pq, fst * snd);
    } else if (operator == '/') {
      push(pq, fst / snd);
    } else {
```

```
60        return 0;
61      }
62      return 1;
63    }
64  }
65
66  // Puts into data the number on top of the stack, and removes the top.
67  // Returns true if everything OK or false in other case.
68  int pop(T_Stack *pq, int *data) {
69    if (*pq == NULL) {
70      return 0;
71    } else {
72      *data = (*pq)->number;
73      *pq = (*pq)->next;
74      return 1;
75    }
76  }
77
78  // Frees the memory of a stack and sets it to empty.
79  void destroy(T_Stack *pq) {
80    while(*pq != NULL) {
81      T_Stack delete = *pq;
82      *pq = (*pq)->next;
83      free(delete);
84    }
85  }
86
87
88  //TERMINA AQUI
```

```c
//EMPIEZA AQUI


#include <stdio.h>
#include <stdlib.h>
#include "Stack.h"

// Creates an empty stack.
T_Stack create() {
  return NULL;
}

// Returns true if the stack is empty and false in other case.
int isEmpty(T_Stack q) {
  return q == NULL ? 1 : 0;
}

// Inserts a number into the stack.
void push(T_Stack *pq, int operand) {
  T_Stack nuevo = (T_Stack) malloc(sizeof(struct Node));
  if (nuevo == NULL) {
    perror("Error al insertar");
  } else {
    nuevo->number = operand;

    if (*pq == NULL) {
      nuevo->next = NULL;
    } else {
      nuevo->next = *pq;
    }
    *pq = nuevo;
  }
}

// "Inserts" an operator into the stack and operates.
// Returns true if everything OK or false in other case.
int pushOperator(T_Stack *pq, char operator) {
  int count2 = 0;
  T_Stack iter = *pq;
  while (iter != NULL && count2 < 2) {
    iter = iter->next;
    count2++;
  }

  if (count2 != 2) {
    return 0;
  } else {
    int fst, snd;
    pop(pq, &snd);
    pop(pq, &fst);
    if (operator == '+') {
      push(pq, fst + snd);
    } else if (operator == '-') {
      push(pq, fst - snd);
    } else if (operator == '*') {
      push(pq, fst * snd);
    } else if (operator == '/') {
      push(pq, fst / snd);
    } else {
```

```
60        return 0;
61      }
62      return 1;
63    }
64  }
65
66  // Puts into data the number on top of the stack, and removes the top.
67  // Returns true if everything OK or false in other case.
68  int pop(T_Stack *pq, int *data) {
69    if (*pq == NULL) {
70      return 0;
71    } else {
72      *data = (*pq)->number;
73      *pq = (*pq)->next;
74      return 1;
75    }
76  }
77
78  // Frees the memory of a stack and sets it to empty.
79  void destroy(T_Stack *pq) {
80    while(*pq != NULL) {
81      T_Stack delete = *pq;
82      *pq = (*pq)->next;
83      free(delete);
84    }
85  }
86
87
88  //TERMINA AQUI
```

```c
//EMPIEZA AQUI


#include <stdio.h>
#include <stdlib.h>
#include "Stack.h"

// Creates an empty stack.
T_Stack create() {
  return NULL;
}

// Returns true if the stack is empty and false in other case.
int isEmpty(T_Stack q) {
  return q == NULL ? 1 : 0;
}

// Inserts a number into the stack.
void push(T_Stack *pq, int operand) {
  T_Stack nuevo = (T_Stack) malloc(sizeof(struct Node));
  if (nuevo == NULL) {
    perror("Error al insertar");
  } else {
    nuevo->number = operand;

    if (*pq == NULL) {
      nuevo->next = NULL;
    } else {
      nuevo->next = *pq;
    }
    *pq = nuevo;
  }
}

// "Inserts" an operator into the stack and operates.
// Returns true if everything OK or false in other case.
int pushOperator(T_Stack *pq, char operator) {
  int count2 = 0;
  T_Stack iter = *pq;
  while (iter != NULL && count2 < 2) {
    iter = iter->next;
    count2++;
  }

  if (count2 != 2) {
    return 0;
  } else {
    int fst, snd;
    pop(pq, &snd);
    pop(pq, &fst);
    if (operator == '+') {
      push(pq, fst + snd);
    } else if (operator == '-') {
      push(pq, fst - snd);
    } else if (operator == '*') {
      push(pq, fst * snd);
    } else if (operator == '/') {
      push(pq, fst / snd);
    } else {
```

```
60        return 0;
61      }
62      return 1;
63    }
64  }
65
66  // Puts into data the number on top of the stack, and removes the top.
67  // Returns true if everything OK or false in other case.
68  int pop(T_Stack *pq, int *data) {
69    if (*pq == NULL) {
70      return 0;
71    } else {
72      *data = (*pq)->number;
73      *pq = (*pq)->next;
74      return 1;
75    }
76  }
77
78  // Frees the memory of a stack and sets it to empty.
79  void destroy(T_Stack *pq) {
80    while(*pq != NULL) {
81      T_Stack delete = *pq;
82      *pq = (*pq)->next;
83      free(delete);
84    }
85  }
86
87
88  //TERMINA AQUI
```

```c
//EMPIEZA AQUI


#include <stdio.h>
#include <stdlib.h>
#include "Stack.h"

// Creates an empty stack.
T_Stack create() {
  return NULL;
}

// Returns true if the stack is empty and false in other case.
int isEmpty(T_Stack q) {
  return q == NULL ? 1 : 0;
}

// Inserts a number into the stack.
void push(T_Stack *pq, int operand) {
  T_Stack nuevo = (T_Stack) malloc(sizeof(struct Node));
  if (nuevo == NULL) {
    perror("Error al insertar");
  } else {
    nuevo->number = operand;

    if (*pq == NULL) {
      nuevo->next = NULL;
    } else {
      nuevo->next = *pq;
    }
    *pq = nuevo;
  }
}

// "Inserts" an operator into the stack and operates.
// Returns true if everything OK or false in other case.
int pushOperator(T_Stack *pq, char operator) {
  int count2 = 0;
  T_Stack iter = *pq;
  while (iter != NULL && count2 < 2) {
    iter = iter->next;
    count2++;
  }

  if (count2 != 2) {
    return 0;
  } else {
    int fst, snd;
    pop(pq, &snd);
    pop(pq, &fst);
    if (operator == '+') {
      push(pq, fst + snd);
    } else if (operator == '-') {
      push(pq, fst - snd);
    } else if (operator == '*') {
      push(pq, fst * snd);
    } else if (operator == '/') {
      push(pq, fst / snd);
    } else {
```

```
60        return 0;
61      }
62      return 1;
63    }
64  }
65
66  // Puts into data the number on top of the stack, and removes the top.
67  // Returns true if everything OK or false in other case.
68  int pop(T_Stack *pq, int *data) {
69    if (*pq == NULL) {
70      return 0;
71    } else {
72      *data = (*pq)->number;
73      *pq = (*pq)->next;
74      return 1;
75    }
76  }
77
78  // Frees the memory of a stack and sets it to empty.
79  void destroy(T_Stack *pq) {
80    while(*pq != NULL) {
81      T_Stack delete = *pq;
82      *pq = (*pq)->next;
83      free(delete);
84    }
85  }
86
87
88  //TERMINA AQUI
```

```c
//EMPIEZA AQUI


#include <stdio.h>
#include <stdlib.h>
#include "Stack.h"

// Creates an empty stack.
T_Stack create() {
  return NULL;
}

// Returns true if the stack is empty and false in other case.
int isEmpty(T_Stack q) {
  return q == NULL ? 1 : 0;
}

// Inserts a number into the stack.
void push(T_Stack *pq, int operand) {
  T_Stack nuevo = (T_Stack) malloc(sizeof(struct Node));
  if (nuevo == NULL) {
    perror("Error al insertar");
  } else {
    nuevo->number = operand;

    if (*pq == NULL) {
      nuevo->next = NULL;
    } else {
      nuevo->next = *pq;
    }
    *pq = nuevo;
  }
}

// "Inserts" an operator into the stack and operates.
// Returns true if everything OK or false in other case.
int pushOperator(T_Stack *pq, char operator) {
  int count2 = 0;
  T_Stack iter = *pq;
  while (iter != NULL && count2 < 2) {
    iter = iter->next;
    count2++;
  }

  if (count2 != 2) {
    return 0;
  } else {
    int fst, snd;
    pop(pq, &snd);
    pop(pq, &fst);
    if (operator == '+') {
      push(pq, fst + snd);
    } else if (operator == '-') {
      push(pq, fst - snd);
    } else if (operator == '*') {
      push(pq, fst * snd);
    } else if (operator == '/') {
      push(pq, fst / snd);
    } else {
```

```
60        return 0;
61      }
62      return 1;
63    }
64  }
65
66  // Puts into data the number on top of the stack, and removes the top.
67  // Returns true if everything OK or false in other case.
68  int pop(T_Stack *pq, int *data) {
69    if (*pq == NULL) {
70      return 0;
71    } else {
72      *data = (*pq)->number;
73      *pq = (*pq)->next;
74      return 1;
75    }
76  }
77
78  // Frees the memory of a stack and sets it to empty.
79  void destroy(T_Stack *pq) {
80    while(*pq != NULL) {
81      T_Stack delete = *pq;
82      *pq = (*pq)->next;
83      free(delete);
84    }
85  }
86
87
88  //TERMINA AQUI
```

```c
//EMPIEZA AQUI


#include <stdio.h>
#include <stdlib.h>
#include "Stack.h"

// Creates an empty stack.
T_Stack create() {
  return NULL;
}

// Returns true if the stack is empty and false in other case.
int isEmpty(T_Stack q) {
  return q == NULL ? 1 : 0;
}

// Inserts a number into the stack.
void push(T_Stack *pq, int operand) {
  T_Stack nuevo = (T_Stack) malloc(sizeof(struct Node));
  if (nuevo == NULL) {
    perror("Error al insertar");
  } else {
    nuevo->number = operand;

    if (*pq == NULL) {
      nuevo->next = NULL;
    } else {
      nuevo->next = *pq;
    }
    *pq = nuevo;
  }
}

// "Inserts" an operator into the stack and operates.
// Returns true if everything OK or false in other case.
int pushOperator(T_Stack *pq, char operator) {
  int count2 = 0;
  T_Stack iter = *pq;
  while (iter != NULL && count2 < 2) {
    iter = iter->next;
    count2++;
  }

  if (count2 != 2) {
    return 0;
  } else {
    int fst, snd;
    pop(pq, &snd);
    pop(pq, &fst);
    if (operator == '+') {
      push(pq, fst + snd);
    } else if (operator == '-') {
      push(pq, fst - snd);
    } else if (operator == '*') {
      push(pq, fst * snd);
    } else if (operator == '/') {
      push(pq, fst / snd);
    } else {
```

```
60        return 0;
61      }
62      return 1;
63    }
64  }
65
66  // Puts into data the number on top of the stack, and removes the top.
67  // Returns true if everything OK or false in other case.
68  int pop(T_Stack *pq, int *data) {
69    if (*pq == NULL) {
70      return 0;
71    } else {
72      *data = (*pq)->number;
73      *pq = (*pq)->next;
74      return 1;
75    }
76  }
77
78  // Frees the memory of a stack and sets it to empty.
79  void destroy(T_Stack *pq) {
80    while(*pq != NULL) {
81      T_Stack delete = *pq;
82      *pq = (*pq)->next;
83      free(delete);
84    }
85  }
86
87
88  //TERMINA AQUI
```

```c
//EMPIEZA AQUI


#include <stdio.h>
#include <stdlib.h>
#include "Stack.h"

// Creates an empty stack.
T_Stack create() {
  return NULL;
}

// Returns true if the stack is empty and false in other case.
int isEmpty(T_Stack q) {
  return q == NULL ? 1 : 0;
}

// Inserts a number into the stack.
void push(T_Stack *pq, int operand) {
  T_Stack nuevo = (T_Stack) malloc(sizeof(struct Node));
  if (nuevo == NULL) {
    perror("Error al insertar");
  } else {
    nuevo->number = operand;

    if (*pq == NULL) {
      nuevo->next = NULL;
    } else {
      nuevo->next = *pq;
    }
    *pq = nuevo;
  }
}

// "Inserts" an operator into the stack and operates.
// Returns true if everything OK or false in other case.
int pushOperator(T_Stack *pq, char operator) {
  int count2 = 0;
  T_Stack iter = *pq;
  while (iter != NULL && count2 < 2) {
    iter = iter->next;
    count2++;
  }

  if (count2 != 2) {
    return 0;
  } else {
    int fst, snd;
    pop(pq, &snd);
    pop(pq, &fst);
    if (operator == '+') {
      push(pq, fst + snd);
    } else if (operator == '-') {
      push(pq, fst - snd);
    } else if (operator == '*') {
      push(pq, fst * snd);
    } else if (operator == '/') {
      push(pq, fst / snd);
    } else {
```

```
60        return 0;
61      }
62      return 1;
63    }
64  }
65
66  // Puts into data the number on top of the stack, and removes the top.
67  // Returns true if everything OK or false in other case.
68  int pop(T_Stack *pq, int *data) {
69    if (*pq == NULL) {
70      return 0;
71    } else {
72      *data = (*pq)->number;
73      *pq = (*pq)->next;
74      return 1;
75    }
76  }
77
78  // Frees the memory of a stack and sets it to empty.
79  void destroy(T_Stack *pq) {
80    while(*pq != NULL) {
81      T_Stack delete = *pq;
82      *pq = (*pq)->next;
83      free(delete);
84    }
85  }
86
87
88  //TERMINA AQUI
```

```c
//EMPIEZA AQUI


#include <stdio.h>
#include <stdlib.h>
#include "Stack.h"

// Creates an empty stack.
T_Stack create() {
  return NULL;
}

// Returns true if the stack is empty and false in other case.
int isEmpty(T_Stack q) {
  return q == NULL ? 1 : 0;
}

// Inserts a number into the stack.
void push(T_Stack *pq, int operand) {
  T_Stack nuevo = (T_Stack) malloc(sizeof(struct Node));
  if (nuevo == NULL) {
    perror("Error al insertar");
  } else {
    nuevo->number = operand;

    if (*pq == NULL) {
      nuevo->next = NULL;
    } else {
      nuevo->next = *pq;
    }
    *pq = nuevo;
  }
}

// "Inserts" an operator into the stack and operates.
// Returns true if everything OK or false in other case.
int pushOperator(T_Stack *pq, char operator) {
  int count2 = 0;
  T_Stack iter = *pq;
  while (iter != NULL && count2 < 2) {
    iter = iter->next;
    count2++;
  }

  if (count2 != 2) {
    return 0;
  } else {
    int fst, snd;
    pop(pq, &snd);
    pop(pq, &fst);
    if (operator == '+') {
      push(pq, fst + snd);
    } else if (operator == '-') {
      push(pq, fst - snd);
    } else if (operator == '*') {
      push(pq, fst * snd);
    } else if (operator == '/') {
      push(pq, fst / snd);
    } else {
```

```
60        return 0;
61      }
62      return 1;
63    }
64  }
65
66  // Puts into data the number on top of the stack, and removes the top.
67  // Returns true if everything OK or false in other case.
68  int pop(T_Stack *pq, int *data) {
69    if (*pq == NULL) {
70      return 0;
71    } else {
72      *data = (*pq)->number;
73      *pq = (*pq)->next;
74      return 1;
75    }
76  }
77
78  // Frees the memory of a stack and sets it to empty.
79  void destroy(T_Stack *pq) {
80    while(*pq != NULL) {
81      T_Stack delete = *pq;
82      *pq = (*pq)->next;
83      free(delete);
84    }
85  }
86
87
88  //TERMINA AQUI
```

```c
//EMPIEZA AQUI


#include <stdio.h>
#include <stdlib.h>
#include "Stack.h"

// Creates an empty stack.
T_Stack create() {
  return NULL;
}

// Returns true if the stack is empty and false in other case.
int isEmpty(T_Stack q) {
  return q == NULL ? 1 : 0;
}

// Inserts a number into the stack.
void push(T_Stack *pq, int operand) {
  T_Stack nuevo = (T_Stack) malloc(sizeof(struct Node));
  if (nuevo == NULL) {
    perror("Error al insertar");
  } else {
    nuevo->number = operand;

    if (*pq == NULL) {
      nuevo->next = NULL;
    } else {
      nuevo->next = *pq;
    }
    *pq = nuevo;
  }
}

// "Inserts" an operator into the stack and operates.
// Returns true if everything OK or false in other case.
int pushOperator(T_Stack *pq, char operator) {
  int count2 = 0;
  T_Stack iter = *pq;
  while (iter != NULL && count2 < 2) {
    iter = iter->next;
    count2++;
  }

  if (count2 != 2) {
    return 0;
  } else {
    int fst, snd;
    pop(pq, &snd);
    pop(pq, &fst);
    if (operator == '+') {
      push(pq, fst + snd);
    } else if (operator == '-') {
      push(pq, fst - snd);
    } else if (operator == '*') {
      push(pq, fst * snd);
    } else if (operator == '/') {
      push(pq, fst / snd);
    } else {
```

```
60        return 0;
61      }
62      return 1;
63    }
64  }
65
66  // Puts into data the number on top of the stack, and removes the top.
67  // Returns true if everything OK or false in other case.
68  int pop(T_Stack *pq, int *data) {
69    if (*pq == NULL) {
70      return 0;
71    } else {
72      *data = (*pq)->number;
73      *pq = (*pq)->next;
74      return 1;
75    }
76  }
77
78  // Frees the memory of a stack and sets it to empty.
79  void destroy(T_Stack *pq) {
80    while(*pq != NULL) {
81      T_Stack delete = *pq;
82      *pq = (*pq)->next;
83      free(delete);
84    }
85  }
86
87
88  //TERMINA AQUI
```

```c
//EMPIEZA AQUI


#include <stdio.h>
#include <stdlib.h>
#include "Stack.h"

// Creates an empty stack.
T_Stack create() {
  return NULL;
}

// Returns true if the stack is empty and false in other case.
int isEmpty(T_Stack q) {
  return q == NULL ? 1 : 0;
}

// Inserts a number into the stack.
void push(T_Stack *pq, int operand) {
  T_Stack nuevo = (T_Stack) malloc(sizeof(struct Node));
  if (nuevo == NULL) {
    perror("Error al insertar");
  } else {
    nuevo->number = operand;

    if (*pq == NULL) {
      nuevo->next = NULL;
    } else {
      nuevo->next = *pq;
    }
    *pq = nuevo;
  }
}

// "Inserts" an operator into the stack and operates.
// Returns true if everything OK or false in other case.
int pushOperator(T_Stack *pq, char operator) {
  int count2 = 0;
  T_Stack iter = *pq;
  while (iter != NULL && count2 < 2) {
    iter = iter->next;
    count2++;
  }

  if (count2 != 2) {
    return 0;
  } else {
    int fst, snd;
    pop(pq, &snd);
    pop(pq, &fst);
    if (operator == '+') {
      push(pq, fst + snd);
    } else if (operator == '-') {
      push(pq, fst - snd);
    } else if (operator == '*') {
      push(pq, fst * snd);
    } else if (operator == '/') {
      push(pq, fst / snd);
    } else {
```

```
60        return 0;
61      }
62      return 1;
63    }
64  }
65
66  // Puts into data the number on top of the stack, and removes the top.
67  // Returns true if everything OK or false in other case.
68  int pop(T_Stack *pq, int *data) {
69    if (*pq == NULL) {
70      return 0;
71    } else {
72      *data = (*pq)->number;
73      *pq = (*pq)->next;
74      return 1;
75    }
76  }
77
78  // Frees the memory of a stack and sets it to empty.
79  void destroy(T_Stack *pq) {
80    while(*pq != NULL) {
81      T_Stack delete = *pq;
82      *pq = (*pq)->next;
83      free(delete);
84    }
85  }
86
87
88  //TERMINA AQUI
```

```c
//EMPIEZA AQUI


#include <stdio.h>
#include <stdlib.h>
#include "Stack.h"

// Creates an empty stack.
T_Stack create() {
  return NULL;
}

// Returns true if the stack is empty and false in other case.
int isEmpty(T_Stack q) {
  return q == NULL ? 1 : 0;
}

// Inserts a number into the stack.
void push(T_Stack *pq, int operand) {
  T_Stack nuevo = (T_Stack) malloc(sizeof(struct Node));
  if (nuevo == NULL) {
    perror("Error al insertar");
  } else {
    nuevo->number = operand;

    if (*pq == NULL) {
      nuevo->next = NULL;
    } else {
      nuevo->next = *pq;
    }
    *pq = nuevo;
  }
}

// "Inserts" an operator into the stack and operates.
// Returns true if everything OK or false in other case.
int pushOperator(T_Stack *pq, char operator) {
  int count2 = 0;
  T_Stack iter = *pq;
  while (iter != NULL && count2 < 2) {
    iter = iter->next;
    count2++;
  }

  if (count2 != 2) {
    return 0;
  } else {
    int fst, snd;
    pop(pq, &snd);
    pop(pq, &fst);
    if (operator == '+') {
      push(pq, fst + snd);
    } else if (operator == '-') {
      push(pq, fst - snd);
    } else if (operator == '*') {
      push(pq, fst * snd);
    } else if (operator == '/') {
      push(pq, fst / snd);
    } else {
```

```
60        return 0;
61      }
62      return 1;
63    }
64  }
65
66  // Puts into data the number on top of the stack, and removes the top.
67  // Returns true if everything OK or false in other case.
68  int pop(T_Stack *pq, int *data) {
69    if (*pq == NULL) {
70      return 0;
71    } else {
72      *data = (*pq)->number;
73      *pq = (*pq)->next;
74      return 1;
75    }
76  }
77
78  // Frees the memory of a stack and sets it to empty.
79  void destroy(T_Stack *pq) {
80    while(*pq != NULL) {
81      T_Stack delete = *pq;
82      *pq = (*pq)->next;
83      free(delete);
84    }
85  }
86
87
88  //TERMINA AQUI
```

```c
//EMPIEZA AQUI


#include <stdio.h>
#include <stdlib.h>
#include "Stack.h"

// Creates an empty stack.
T_Stack create() {
  return NULL;
}

// Returns true if the stack is empty and false in other case.
int isEmpty(T_Stack q) {
  return q == NULL ? 1 : 0;
}

// Inserts a number into the stack.
void push(T_Stack *pq, int operand) {
  T_Stack nuevo = (T_Stack) malloc(sizeof(struct Node));
  if (nuevo == NULL) {
    perror("Error al insertar");
  } else {
    nuevo->number = operand;

    if (*pq == NULL) {
      nuevo->next = NULL;
    } else {
      nuevo->next = *pq;
    }
    *pq = nuevo;
  }
}

// "Inserts" an operator into the stack and operates.
// Returns true if everything OK or false in other case.
int pushOperator(T_Stack *pq, char operator) {
  int count2 = 0;
  T_Stack iter = *pq;
  while (iter != NULL && count2 < 2) {
    iter = iter->next;
    count2++;
  }

  if (count2 != 2) {
    return 0;
  } else {
    int fst, snd;
    pop(pq, &snd);
    pop(pq, &fst);
    if (operator == '+') {
      push(pq, fst + snd);
    } else if (operator == '-') {
      push(pq, fst - snd);
    } else if (operator == '*') {
      push(pq, fst * snd);
    } else if (operator == '/') {
      push(pq, fst / snd);
    } else {
```

```
60        return 0;
61      }
62      return 1;
63    }
64  }
65
66  // Puts into data the number on top of the stack, and removes the top.
67  // Returns true if everything OK or false in other case.
68  int pop(T_Stack *pq, int *data) {
69    if (*pq == NULL) {
70      return 0;
71    } else {
72      *data = (*pq)->number;
73      *pq = (*pq)->next;
74      return 1;
75    }
76  }
77
78  // Frees the memory of a stack and sets it to empty.
79  void destroy(T_Stack *pq) {
80    while(*pq != NULL) {
81      T_Stack delete = *pq;
82      *pq = (*pq)->next;
83      free(delete);
84    }
85  }
86
87
88  //TERMINA AQUI
```

```c
//EMPIEZA AQUI


#include <stdio.h>
#include <stdlib.h>
#include "Stack.h"

// Creates an empty stack.
T_Stack create() {
  return NULL;
}

// Returns true if the stack is empty and false in other case.
int isEmpty(T_Stack q) {
  return q == NULL ? 1 : 0;
}

// Inserts a number into the stack.
void push(T_Stack *pq, int operand) {
  T_Stack nuevo = (T_Stack) malloc(sizeof(struct Node));
  if (nuevo == NULL) {
    perror("Error al insertar");
  } else {
    nuevo->number = operand;

    if (*pq == NULL) {
      nuevo->next = NULL;
    } else {
      nuevo->next = *pq;
    }
    *pq = nuevo;
  }
}

// "Inserts" an operator into the stack and operates.
// Returns true if everything OK or false in other case.
int pushOperator(T_Stack *pq, char operator) {
  int count2 = 0;
  T_Stack iter = *pq;
  while (iter != NULL && count2 < 2) {
    iter = iter->next;
    count2++;
  }

  if (count2 != 2) {
    return 0;
  } else {
    int fst, snd;
    pop(pq, &snd);
    pop(pq, &fst);
    if (operator == '+') {
      push(pq, fst + snd);
    } else if (operator == '-') {
      push(pq, fst - snd);
    } else if (operator == '*') {
      push(pq, fst * snd);
    } else if (operator == '/') {
      push(pq, fst / snd);
    } else {
```

```
60        return 0;
61      }
62      return 1;
63    }
64  }
65
66  // Puts into data the number on top of the stack, and removes the top.
67  // Returns true if everything OK or false in other case.
68  int pop(T_Stack *pq, int *data) {
69    if (*pq == NULL) {
70      return 0;
71    } else {
72      *data = (*pq)->number;
73      *pq = (*pq)->next;
74      return 1;
75    }
76  }
77
78  // Frees the memory of a stack and sets it to empty.
79  void destroy(T_Stack *pq) {
80    while(*pq != NULL) {
81      T_Stack delete = *pq;
82      *pq = (*pq)->next;
83      free(delete);
84    }
85  }
86
87
88  //TERMINA AQUI
```

```c
//EMPIEZA AQUI


#include <stdio.h>
#include <stdlib.h>
#include "Stack.h"

// Creates an empty stack.
T_Stack create() {
  return NULL;
}

// Returns true if the stack is empty and false in other case.
int isEmpty(T_Stack q) {
  return q == NULL ? 1 : 0;
}

// Inserts a number into the stack.
void push(T_Stack *pq, int operand) {
  T_Stack nuevo = (T_Stack) malloc(sizeof(struct Node));
  if (nuevo == NULL) {
    perror("Error al insertar");
  } else {
    nuevo->number = operand;

    if (*pq == NULL) {
      nuevo->next = NULL;
    } else {
      nuevo->next = *pq;
    }
    *pq = nuevo;
  }
}

// "Inserts" an operator into the stack and operates.
// Returns true if everything OK or false in other case.
int pushOperator(T_Stack *pq, char operator) {
  int count2 = 0;
  T_Stack iter = *pq;
  while (iter != NULL && count2 < 2) {
    iter = iter->next;
    count2++;
  }

  if (count2 != 2) {
    return 0;
  } else {
    int fst, snd;
    pop(pq, &snd);
    pop(pq, &fst);
    if (operator == '+') {
      push(pq, fst + snd);
    } else if (operator == '-') {
      push(pq, fst - snd);
    } else if (operator == '*') {
      push(pq, fst * snd);
    } else if (operator == '/') {
      push(pq, fst / snd);
    } else {
```

```
60        return 0;
61      }
62      return 1;
63    }
64  }
65
66  // Puts into data the number on top of the stack, and removes the top.
67  // Returns true if everything OK or false in other case.
68  int pop(T_Stack *pq, int *data) {
69    if (*pq == NULL) {
70      return 0;
71    } else {
72      *data = (*pq)->number;
73      *pq = (*pq)->next;
74      return 1;
75    }
76  }
77
78  // Frees the memory of a stack and sets it to empty.
79  void destroy(T_Stack *pq) {
80    while(*pq != NULL) {
81      T_Stack delete = *pq;
82      *pq = (*pq)->next;
83      free(delete);
84    }
85  }
86
87
88  //TERMINA AQUI
```

```c
//EMPIEZA AQUI


#include <stdio.h>
#include <stdlib.h>
#include "Stack.h"

// Creates an empty stack.
T_Stack create() {
  return NULL;
}

// Returns true if the stack is empty and false in other case.
int isEmpty(T_Stack q) {
  return q == NULL ? 1 : 0;
}

// Inserts a number into the stack.
void push(T_Stack *pq, int operand) {
  T_Stack nuevo = (T_Stack) malloc(sizeof(struct Node));
  if (nuevo == NULL) {
    perror("Error al insertar");
  } else {
    nuevo->number = operand;

    if (*pq == NULL) {
      nuevo->next = NULL;
    } else {
      nuevo->next = *pq;
    }
    *pq = nuevo;
  }
}

// "Inserts" an operator into the stack and operates.
// Returns true if everything OK or false in other case.
int pushOperator(T_Stack *pq, char operator) {
  int count2 = 0;
  T_Stack iter = *pq;
  while (iter != NULL && count2 < 2) {
    iter = iter->next;
    count2++;
  }

  if (count2 != 2) {
    return 0;
  } else {
    int fst, snd;
    pop(pq, &snd);
    pop(pq, &fst);
    if (operator == '+') {
      push(pq, fst + snd);
    } else if (operator == '-') {
      push(pq, fst - snd);
    } else if (operator == '*') {
      push(pq, fst * snd);
    } else if (operator == '/') {
      push(pq, fst / snd);
    } else {
```

```
60        return 0;
61      }
62      return 1;
63    }
64  }
65
66  // Puts into data the number on top of the stack, and removes the top.
67  // Returns true if everything OK or false in other case.
68  int pop(T_Stack *pq, int *data) {
69    if (*pq == NULL) {
70      return 0;
71    } else {
72      *data = (*pq)->number;
73      *pq = (*pq)->next;
74      return 1;
75    }
76  }
77
78  // Frees the memory of a stack and sets it to empty.
79  void destroy(T_Stack *pq) {
80    while(*pq != NULL) {
81      T_Stack delete = *pq;
82      *pq = (*pq)->next;
83      free(delete);
84    }
85  }
86
87
88  //TERMINA AQUI
```

```c
//EMPIEZA AQUI


#include <stdio.h>
#include <stdlib.h>
#include "Stack.h"

// Creates an empty stack.
T_Stack create() {
  return NULL;
}

// Returns true if the stack is empty and false in other case.
int isEmpty(T_Stack q) {
  return q == NULL ? 1 : 0;
}

// Inserts a number into the stack.
void push(T_Stack *pq, int operand) {
  T_Stack nuevo = (T_Stack) malloc(sizeof(struct Node));
  if (nuevo == NULL) {
    perror("Error al insertar");
  } else {
    nuevo->number = operand;

    if (*pq == NULL) {
      nuevo->next = NULL;
    } else {
      nuevo->next = *pq;
    }
    *pq = nuevo;
  }
}

// "Inserts" an operator into the stack and operates.
// Returns true if everything OK or false in other case.
int pushOperator(T_Stack *pq, char operator) {
  int count2 = 0;
  T_Stack iter = *pq;
  while (iter != NULL && count2 < 2) {
    iter = iter->next;
    count2++;
  }

  if (count2 != 2) {
    return 0;
  } else {
    int fst, snd;
    pop(pq, &snd);
    pop(pq, &fst);
    if (operator == '+') {
      push(pq, fst + snd);
    } else if (operator == '-') {
      push(pq, fst - snd);
    } else if (operator == '*') {
      push(pq, fst * snd);
    } else if (operator == '/') {
      push(pq, fst / snd);
    } else {
```

```
60        return 0;
61      }
62      return 1;
63    }
64  }
65
66  // Puts into data the number on top of the stack, and removes the top.
67  // Returns true if everything OK or false in other case.
68  int pop(T_Stack *pq, int *data) {
69    if (*pq == NULL) {
70      return 0;
71    } else {
72      *data = (*pq)->number;
73      *pq = (*pq)->next;
74      return 1;
75    }
76  }
77
78  // Frees the memory of a stack and sets it to empty.
79  void destroy(T_Stack *pq) {
80    while(*pq != NULL) {
81      T_Stack delete = *pq;
82      *pq = (*pq)->next;
83      free(delete);
84    }
85  }
86
87
88  //TERMINA AQUI
```

```c
//EMPIEZA AQUI


#include <stdio.h>
#include <stdlib.h>
#include "Stack.h"

// Creates an empty stack.
T_Stack create() {
  return NULL;
}

// Returns true if the stack is empty and false in other case.
int isEmpty(T_Stack q) {
  return q == NULL ? 1 : 0;
}

// Inserts a number into the stack.
void push(T_Stack *pq, int operand) {
  T_Stack nuevo = (T_Stack) malloc(sizeof(struct Node));
  if (nuevo == NULL) {
    perror("Error al insertar");
  } else {
    nuevo->number = operand;

    if (*pq == NULL) {
      nuevo->next = NULL;
    } else {
      nuevo->next = *pq;
    }
    *pq = nuevo;
  }
}

// "Inserts" an operator into the stack and operates.
// Returns true if everything OK or false in other case.
int pushOperator(T_Stack *pq, char operator) {
  int count2 = 0;
  T_Stack iter = *pq;
  while (iter != NULL && count2 < 2) {
    iter = iter->next;
    count2++;
  }

  if (count2 != 2) {
    return 0;
  } else {
    int fst, snd;
    pop(pq, &snd);
    pop(pq, &fst);
    if (operator == '+') {
      push(pq, fst + snd);
    } else if (operator == '-') {
      push(pq, fst - snd);
    } else if (operator == '*') {
      push(pq, fst * snd);
    } else if (operator == '/') {
      push(pq, fst / snd);
    } else {
```

```
60        return 0;
61      }
62      return 1;
63    }
64  }
65
66  // Puts into data the number on top of the stack, and removes the top.
67  // Returns true if everything OK or false in other case.
68  int pop(T_Stack *pq, int *data) {
69    if (*pq == NULL) {
70      return 0;
71    } else {
72      *data = (*pq)->number;
73      *pq = (*pq)->next;
74      return 1;
75    }
76  }
77
78  // Frees the memory of a stack and sets it to empty.
79  void destroy(T_Stack *pq) {
80    while(*pq != NULL) {
81      T_Stack delete = *pq;
82      *pq = (*pq)->next;
83      free(delete);
84    }
85  }
86
87
88  //TERMINA AQUI
```

```c
//EMPIEZA AQUI


#include <stdio.h>
#include <stdlib.h>
#include "Stack.h"

// Creates an empty stack.
T_Stack create() {
  return NULL;
}

// Returns true if the stack is empty and false in other case.
int isEmpty(T_Stack q) {
  return q == NULL ? 1 : 0;
}

// Inserts a number into the stack.
void push(T_Stack *pq, int operand) {
  T_Stack nuevo = (T_Stack) malloc(sizeof(struct Node));
  if (nuevo == NULL) {
    perror("Error al insertar");
  } else {
    nuevo->number = operand;

    if (*pq == NULL) {
      nuevo->next = NULL;
    } else {
      nuevo->next = *pq;
    }
    *pq = nuevo;
  }
}

// "Inserts" an operator into the stack and operates.
// Returns true if everything OK or false in other case.
int pushOperator(T_Stack *pq, char operator) {
  int count2 = 0;
  T_Stack iter = *pq;
  while (iter != NULL && count2 < 2) {
    iter = iter->next;
    count2++;
  }

  if (count2 != 2) {
    return 0;
  } else {
    int fst, snd;
    pop(pq, &snd);
    pop(pq, &fst);
    if (operator == '+') {
      push(pq, fst + snd);
    } else if (operator == '-') {
      push(pq, fst - snd);
    } else if (operator == '*') {
      push(pq, fst * snd);
    } else if (operator == '/') {
      push(pq, fst / snd);
    } else {
```

```
60        return 0;
61      }
62      return 1;
63    }
64  }
65
66  // Puts into data the number on top of the stack, and removes the top.
67  // Returns true if everything OK or false in other case.
68  int pop(T_Stack *pq, int *data) {
69    if (*pq == NULL) {
70      return 0;
71    } else {
72      *data = (*pq)->number;
73      *pq = (*pq)->next;
74      return 1;
75    }
76  }
77
78  // Frees the memory of a stack and sets it to empty.
79  void destroy(T_Stack *pq) {
80    while(*pq != NULL) {
81      T_Stack delete = *pq;
82      *pq = (*pq)->next;
83      free(delete);
84    }
85  }
86
87
88  //TERMINA AQUI
```

```c
//EMPIEZA AQUI


#include <stdio.h>
#include <stdlib.h>
#include "Stack.h"

// Creates an empty stack.
T_Stack create() {
  return NULL;
}

// Returns true if the stack is empty and false in other case.
int isEmpty(T_Stack q) {
  return q == NULL ? 1 : 0;
}

// Inserts a number into the stack.
void push(T_Stack *pq, int operand) {
  T_Stack nuevo = (T_Stack) malloc(sizeof(struct Node));
  if (nuevo == NULL) {
    perror("Error al insertar");
  } else {
    nuevo->number = operand;

    if (*pq == NULL) {
      nuevo->next = NULL;
    } else {
      nuevo->next = *pq;
    }
    *pq = nuevo;
  }
}

// "Inserts" an operator into the stack and operates.
// Returns true if everything OK or false in other case.
int pushOperator(T_Stack *pq, char operator) {
  int count2 = 0;
  T_Stack iter = *pq;
  while (iter != NULL && count2 < 2) {
    iter = iter->next;
    count2++;
  }

  if (count2 != 2) {
    return 0;
  } else {
    int fst, snd;
    pop(pq, &snd);
    pop(pq, &fst);
    if (operator == '+') {
      push(pq, fst + snd);
    } else if (operator == '-') {
      push(pq, fst - snd);
    } else if (operator == '*') {
      push(pq, fst * snd);
    } else if (operator == '/') {
      push(pq, fst / snd);
    } else {
```

```
60        return 0;
61      }
62      return 1;
63    }
64  }
65
66  // Puts into data the number on top of the stack, and removes the top.
67  // Returns true if everything OK or false in other case.
68  int pop(T_Stack *pq, int *data) {
69    if (*pq == NULL) {
70      return 0;
71    } else {
72      *data = (*pq)->number;
73      *pq = (*pq)->next;
74      return 1;
75    }
76  }
77
78  // Frees the memory of a stack and sets it to empty.
79  void destroy(T_Stack *pq) {
80    while(*pq != NULL) {
81      T_Stack delete = *pq;
82      *pq = (*pq)->next;
83      free(delete);
84    }
85  }
86
87
88  //TERMINA AQUI
```

```c
//EMPIEZA AQUI


#include <stdio.h>
#include <stdlib.h>
#include "Stack.h"

// Creates an empty stack.
T_Stack create() {
  return NULL;
}

// Returns true if the stack is empty and false in other case.
int isEmpty(T_Stack q) {
  return q == NULL ? 1 : 0;
}

// Inserts a number into the stack.
void push(T_Stack *pq, int operand) {
  T_Stack nuevo = (T_Stack) malloc(sizeof(struct Node));
  if (nuevo == NULL) {
    perror("Error al insertar");
  } else {
    nuevo->number = operand;

    if (*pq == NULL) {
      nuevo->next = NULL;
    } else {
      nuevo->next = *pq;
    }
    *pq = nuevo;
  }
}

// "Inserts" an operator into the stack and operates.
// Returns true if everything OK or false in other case.
int pushOperator(T_Stack *pq, char operator) {
  int count2 = 0;
  T_Stack iter = *pq;
  while (iter != NULL && count2 < 2) {
    iter = iter->next;
    count2++;
  }

  if (count2 != 2) {
    return 0;
  } else {
    int fst, snd;
    pop(pq, &snd);
    pop(pq, &fst);
    if (operator == '+') {
      push(pq, fst + snd);
    } else if (operator == '-') {
      push(pq, fst - snd);
    } else if (operator == '*') {
      push(pq, fst * snd);
    } else if (operator == '/') {
      push(pq, fst / snd);
    } else {
```

```
60        return 0;
61      }
62      return 1;
63    }
64  }
65
66  // Puts into data the number on top of the stack, and removes the top.
67  // Returns true if everything OK or false in other case.
68  int pop(T_Stack *pq, int *data) {
69    if (*pq == NULL) {
70      return 0;
71    } else {
72      *data = (*pq)->number;
73      *pq = (*pq)->next;
74      return 1;
75    }
76  }
77
78  // Frees the memory of a stack and sets it to empty.
79  void destroy(T_Stack *pq) {
80    while(*pq != NULL) {
81      T_Stack delete = *pq;
82      *pq = (*pq)->next;
83      free(delete);
84    }
85  }
86
87
88  //TERMINA AQUI
```

```c
//EMPIEZA AQUI


#include <stdio.h>
#include <stdlib.h>
#include "Stack.h"

// Creates an empty stack.
T_Stack create() {
  return NULL;
}

// Returns true if the stack is empty and false in other case.
int isEmpty(T_Stack q) {
  return q == NULL ? 1 : 0;
}

// Inserts a number into the stack.
void push(T_Stack *pq, int operand) {
  T_Stack nuevo = (T_Stack) malloc(sizeof(struct Node));
  if (nuevo == NULL) {
    perror("Error al insertar");
  } else {
    nuevo->number = operand;

    if (*pq == NULL) {
      nuevo->next = NULL;
    } else {
      nuevo->next = *pq;
    }
    *pq = nuevo;
  }
}

// "Inserts" an operator into the stack and operates.
// Returns true if everything OK or false in other case.
int pushOperator(T_Stack *pq, char operator) {
  int count2 = 0;
  T_Stack iter = *pq;
  while (iter != NULL && count2 < 2) {
    iter = iter->next;
    count2++;
  }

  if (count2 != 2) {
    return 0;
  } else {
    int fst, snd;
    pop(pq, &snd);
    pop(pq, &fst);
    if (operator == '+') {
      push(pq, fst + snd);
    } else if (operator == '-') {
      push(pq, fst - snd);
    } else if (operator == '*') {
      push(pq, fst * snd);
    } else if (operator == '/') {
      push(pq, fst / snd);
    } else {
```

```
60        return 0;
61      }
62      return 1;
63    }
64  }
65
66  // Puts into data the number on top of the stack, and removes the top.
67  // Returns true if everything OK or false in other case.
68  int pop(T_Stack *pq, int *data) {
69    if (*pq == NULL) {
70      return 0;
71    } else {
72      *data = (*pq)->number;
73      *pq = (*pq)->next;
74      return 1;
75    }
76  }
77
78  // Frees the memory of a stack and sets it to empty.
79  void destroy(T_Stack *pq) {
80    while(*pq != NULL) {
81      T_Stack delete = *pq;
82      *pq = (*pq)->next;
83      free(delete);
84    }
85  }
86
87
88  //TERMINA AQUI
```

```c
//EMPIEZA AQUI


#include <stdio.h>
#include <stdlib.h>
#include "Stack.h"

// Creates an empty stack.
T_Stack create() {
  return NULL;
}

// Returns true if the stack is empty and false in other case.
int isEmpty(T_Stack q) {
  return q == NULL ? 1 : 0;
}

// Inserts a number into the stack.
void push(T_Stack *pq, int operand) {
  T_Stack nuevo = (T_Stack) malloc(sizeof(struct Node));
  if (nuevo == NULL) {
    perror("Error al insertar");
  } else {
    nuevo->number = operand;

    if (*pq == NULL) {
      nuevo->next = NULL;
    } else {
      nuevo->next = *pq;
    }
    *pq = nuevo;
  }
}

// "Inserts" an operator into the stack and operates.
// Returns true if everything OK or false in other case.
int pushOperator(T_Stack *pq, char operator) {
  int count2 = 0;
  T_Stack iter = *pq;
  while (iter != NULL && count2 < 2) {
    iter = iter->next;
    count2++;
  }

  if (count2 != 2) {
    return 0;
  } else {
    int fst, snd;
    pop(pq, &snd);
    pop(pq, &fst);
    if (operator == '+') {
      push(pq, fst + snd);
    } else if (operator == '-') {
      push(pq, fst - snd);
    } else if (operator == '*') {
      push(pq, fst * snd);
    } else if (operator == '/') {
      push(pq, fst / snd);
    } else {
```

```
60        return 0;
61      }
62      return 1;
63    }
64  }
65
66  // Puts into data the number on top of the stack, and removes the top.
67  // Returns true if everything OK or false in other case.
68  int pop(T_Stack *pq, int *data) {
69    if (*pq == NULL) {
70      return 0;
71    } else {
72      *data = (*pq)->number;
73      *pq = (*pq)->next;
74      return 1;
75    }
76  }
77
78  // Frees the memory of a stack and sets it to empty.
79  void destroy(T_Stack *pq) {
80    while(*pq != NULL) {
81      T_Stack delete = *pq;
82      *pq = (*pq)->next;
83      free(delete);
84    }
85  }
86
87
88  //TERMINA AQUI
```

```c
//EMPIEZA AQUI


#include <stdio.h>
#include <stdlib.h>
#include "Stack.h"

// Creates an empty stack.
T_Stack create() {
  return NULL;
}

// Returns true if the stack is empty and false in other case.
int isEmpty(T_Stack q) {
  return q == NULL ? 1 : 0;
}

// Inserts a number into the stack.
void push(T_Stack *pq, int operand) {
  T_Stack nuevo = (T_Stack) malloc(sizeof(struct Node));
  if (nuevo == NULL) {
    perror("Error al insertar");
  } else {
    nuevo->number = operand;

    if (*pq == NULL) {
      nuevo->next = NULL;
    } else {
      nuevo->next = *pq;
    }
    *pq = nuevo;
  }
}

// "Inserts" an operator into the stack and operates.
// Returns true if everything OK or false in other case.
int pushOperator(T_Stack *pq, char operator) {
  int count2 = 0;
  T_Stack iter = *pq;
  while (iter != NULL && count2 < 2) {
    iter = iter->next;
    count2++;
  }

  if (count2 != 2) {
    return 0;
  } else {
    int fst, snd;
    pop(pq, &snd);
    pop(pq, &fst);
    if (operator == '+') {
      push(pq, fst + snd);
    } else if (operator == '-') {
      push(pq, fst - snd);
    } else if (operator == '*') {
      push(pq, fst * snd);
    } else if (operator == '/') {
      push(pq, fst / snd);
    } else {
```

```
60        return 0;
61      }
62      return 1;
63    }
64  }
65
66  // Puts into data the number on top of the stack, and removes the top.
67  // Returns true if everything OK or false in other case.
68  int pop(T_Stack *pq, int *data) {
69    if (*pq == NULL) {
70      return 0;
71    } else {
72      *data = (*pq)->number;
73      *pq = (*pq)->next;
74      return 1;
75    }
76  }
77
78  // Frees the memory of a stack and sets it to empty.
79  void destroy(T_Stack *pq) {
80    while(*pq != NULL) {
81      T_Stack delete = *pq;
82      *pq = (*pq)->next;
83      free(delete);
84    }
85  }
86
87
88  //TERMINA AQUI
```

```c
//EMPIEZA AQUI


#include <stdio.h>
#include <stdlib.h>
#include "Stack.h"

// Creates an empty stack.
T_Stack create() {
  return NULL;
}

// Returns true if the stack is empty and false in other case.
int isEmpty(T_Stack q) {
  return q == NULL ? 1 : 0;
}

// Inserts a number into the stack.
void push(T_Stack *pq, int operand) {
  T_Stack nuevo = (T_Stack) malloc(sizeof(struct Node));
  if (nuevo == NULL) {
    perror("Error al insertar");
  } else {
    nuevo->number = operand;

    if (*pq == NULL) {
      nuevo->next = NULL;
    } else {
      nuevo->next = *pq;
    }
    *pq = nuevo;
  }
}

// "Inserts" an operator into the stack and operates.
// Returns true if everything OK or false in other case.
int pushOperator(T_Stack *pq, char operator) {
  int count2 = 0;
  T_Stack iter = *pq;
  while (iter != NULL && count2 < 2) {
    iter = iter->next;
    count2++;
  }

  if (count2 != 2) {
    return 0;
  } else {
    int fst, snd;
    pop(pq, &snd);
    pop(pq, &fst);
    if (operator == '+') {
      push(pq, fst + snd);
    } else if (operator == '-') {
      push(pq, fst - snd);
    } else if (operator == '*') {
      push(pq, fst * snd);
    } else if (operator == '/') {
      push(pq, fst / snd);
    } else {
```

```
60        return 0;
61      }
62      return 1;
63    }
64  }
65
66  // Puts into data the number on top of the stack, and removes the top.
67  // Returns true if everything OK or false in other case.
68  int pop(T_Stack *pq, int *data) {
69    if (*pq == NULL) {
70      return 0;
71    } else {
72      *data = (*pq)->number;
73      *pq = (*pq)->next;
74      return 1;
75    }
76  }
77
78  // Frees the memory of a stack and sets it to empty.
79  void destroy(T_Stack *pq) {
80    while(*pq != NULL) {
81      T_Stack delete = *pq;
82      *pq = (*pq)->next;
83      free(delete);
84    }
85  }
86
87
88  //TERMINA AQUI
```

```c
//EMPIEZA AQUI


#include <stdio.h>
#include <stdlib.h>
#include "Stack.h"

// Creates an empty stack.
T_Stack create() {
  return NULL;
}

// Returns true if the stack is empty and false in other case.
int isEmpty(T_Stack q) {
  return q == NULL ? 1 : 0;
}

// Inserts a number into the stack.
void push(T_Stack *pq, int operand) {
  T_Stack nuevo = (T_Stack) malloc(sizeof(struct Node));
  if (nuevo == NULL) {
    perror("Error al insertar");
  } else {
    nuevo->number = operand;

    if (*pq == NULL) {
      nuevo->next = NULL;
    } else {
      nuevo->next = *pq;
    }
    *pq = nuevo;
  }
}

// "Inserts" an operator into the stack and operates.
// Returns true if everything OK or false in other case.
int pushOperator(T_Stack *pq, char operator) {
  int count2 = 0;
  T_Stack iter = *pq;
  while (iter != NULL && count2 < 2) {
    iter = iter->next;
    count2++;
  }

  if (count2 != 2) {
    return 0;
  } else {
    int fst, snd;
    pop(pq, &snd);
    pop(pq, &fst);
    if (operator == '+') {
      push(pq, fst + snd);
    } else if (operator == '-') {
      push(pq, fst - snd);
    } else if (operator == '*') {
      push(pq, fst * snd);
    } else if (operator == '/') {
      push(pq, fst / snd);
    } else {
```

```
60        return 0;
61      }
62      return 1;
63    }
64  }
65
66  // Puts into data the number on top of the stack, and removes the top.
67  // Returns true if everything OK or false in other case.
68  int pop(T_Stack *pq, int *data) {
69    if (*pq == NULL) {
70      return 0;
71    } else {
72      *data = (*pq)->number;
73      *pq = (*pq)->next;
74      return 1;
75    }
76  }
77
78  // Frees the memory of a stack and sets it to empty.
79  void destroy(T_Stack *pq) {
80    while(*pq != NULL) {
81      T_Stack delete = *pq;
82      *pq = (*pq)->next;
83      free(delete);
84    }
85  }
86
87
88  //TERMINA AQUI
```

```c
//EMPIEZA AQUI


#include <stdio.h>
#include <stdlib.h>
#include "Stack.h"

// Creates an empty stack.
T_Stack create() {
  return NULL;
}

// Returns true if the stack is empty and false in other case.
int isEmpty(T_Stack q) {
  return q == NULL ? 1 : 0;
}

// Inserts a number into the stack.
void push(T_Stack *pq, int operand) {
  T_Stack nuevo = (T_Stack) malloc(sizeof(struct Node));
  if (nuevo == NULL) {
    perror("Error al insertar");
  } else {
    nuevo->number = operand;

    if (*pq == NULL) {
      nuevo->next = NULL;
    } else {
      nuevo->next = *pq;
    }
    *pq = nuevo;
  }
}

// "Inserts" an operator into the stack and operates.
// Returns true if everything OK or false in other case.
int pushOperator(T_Stack *pq, char operator) {
  int count2 = 0;
  T_Stack iter = *pq;
  while (iter != NULL && count2 < 2) {
    iter = iter->next;
    count2++;
  }

  if (count2 != 2) {
    return 0;
  } else {
    int fst, snd;
    pop(pq, &snd);
    pop(pq, &fst);
    if (operator == '+') {
      push(pq, fst + snd);
    } else if (operator == '-') {
      push(pq, fst - snd);
    } else if (operator == '*') {
      push(pq, fst * snd);
    } else if (operator == '/') {
      push(pq, fst / snd);
    } else {
```

```
60        return 0;
61      }
62      return 1;
63    }
64  }
65
66  // Puts into data the number on top of the stack, and removes the top.
67  // Returns true if everything OK or false in other case.
68  int pop(T_Stack *pq, int *data) {
69    if (*pq == NULL) {
70      return 0;
71    } else {
72      *data = (*pq)->number;
73      *pq = (*pq)->next;
74      return 1;
75    }
76  }
77
78  // Frees the memory of a stack and sets it to empty.
79  void destroy(T_Stack *pq) {
80    while(*pq != NULL) {
81      T_Stack delete = *pq;
82      *pq = (*pq)->next;
83      free(delete);
84    }
85  }
86
87
88  //TERMINA AQUI
```

```c
//EMPIEZA AQUI


#include <stdio.h>
#include <stdlib.h>
#include "Stack.h"

// Creates an empty stack.
T_Stack create() {
  return NULL;
}

// Returns true if the stack is empty and false in other case.
int isEmpty(T_Stack q) {
  return q == NULL ? 1 : 0;
}

// Inserts a number into the stack.
void push(T_Stack *pq, int operand) {
  T_Stack nuevo = (T_Stack) malloc(sizeof(struct Node));
  if (nuevo == NULL) {
    perror("Error al insertar");
  } else {
    nuevo->number = operand;

    if (*pq == NULL) {
      nuevo->next = NULL;
    } else {
      nuevo->next = *pq;
    }
    *pq = nuevo;
  }
}

// "Inserts" an operator into the stack and operates.
// Returns true if everything OK or false in other case.
int pushOperator(T_Stack *pq, char operator) {
  int count2 = 0;
  T_Stack iter = *pq;
  while (iter != NULL && count2 < 2) {
    iter = iter->next;
    count2++;
  }

  if (count2 != 2) {
    return 0;
  } else {
    int fst, snd;
    pop(pq, &snd);
    pop(pq, &fst);
    if (operator == '+') {
      push(pq, fst + snd);
    } else if (operator == '-') {
      push(pq, fst - snd);
    } else if (operator == '*') {
      push(pq, fst * snd);
    } else if (operator == '/') {
      push(pq, fst / snd);
    } else {
```

```
60        return 0;
61      }
62      return 1;
63    }
64  }
65
66  // Puts into data the number on top of the stack, and removes the top.
67  // Returns true if everything OK or false in other case.
68  int pop(T_Stack *pq, int *data) {
69    if (*pq == NULL) {
70      return 0;
71    } else {
72      *data = (*pq)->number;
73      *pq = (*pq)->next;
74      return 1;
75    }
76  }
77
78  // Frees the memory of a stack and sets it to empty.
79  void destroy(T_Stack *pq) {
80    while(*pq != NULL) {
81      T_Stack delete = *pq;
82      *pq = (*pq)->next;
83      free(delete);
84    }
85  }
86
87
88  //TERMINA AQUI
```

```c
//EMPIEZA AQUI


#include <stdio.h>
#include <stdlib.h>
#include "Stack.h"

// Creates an empty stack.
T_Stack create() {
  return NULL;
}

// Returns true if the stack is empty and false in other case.
int isEmpty(T_Stack q) {
  return q == NULL ? 1 : 0;
}

// Inserts a number into the stack.
void push(T_Stack *pq, int operand) {
  T_Stack nuevo = (T_Stack) malloc(sizeof(struct Node));
  if (nuevo == NULL) {
    perror("Error al insertar");
  } else {
    nuevo->number = operand;

    if (*pq == NULL) {
      nuevo->next = NULL;
    } else {
      nuevo->next = *pq;
    }
    *pq = nuevo;
  }
}

// "Inserts" an operator into the stack and operates.
// Returns true if everything OK or false in other case.
int pushOperator(T_Stack *pq, char operator) {
  int count2 = 0;
  T_Stack iter = *pq;
  while (iter != NULL && count2 < 2) {
    iter = iter->next;
    count2++;
  }

  if (count2 != 2) {
    return 0;
  } else {
    int fst, snd;
    pop(pq, &snd);
    pop(pq, &fst);
    if (operator == '+') {
      push(pq, fst + snd);
    } else if (operator == '-') {
      push(pq, fst - snd);
    } else if (operator == '*') {
      push(pq, fst * snd);
    } else if (operator == '/') {
      push(pq, fst / snd);
    } else {
```

```
60        return 0;
61      }
62      return 1;
63    }
64  }
65
66  // Puts into data the number on top of the stack, and removes the top.
67  // Returns true if everything OK or false in other case.
68  int pop(T_Stack *pq, int *data) {
69    if (*pq == NULL) {
70      return 0;
71    } else {
72      *data = (*pq)->number;
73      *pq = (*pq)->next;
74      return 1;
75    }
76  }
77
78  // Frees the memory of a stack and sets it to empty.
79  void destroy(T_Stack *pq) {
80    while(*pq != NULL) {
81      T_Stack delete = *pq;
82      *pq = (*pq)->next;
83      free(delete);
84    }
85  }
86
87
88  //TERMINA AQUI
```

```c
//EMPIEZA AQUI


#include <stdio.h>
#include <stdlib.h>
#include "Stack.h"

// Creates an empty stack.
T_Stack create() {
  return NULL;
}

// Returns true if the stack is empty and false in other case.
int isEmpty(T_Stack q) {
  return q == NULL ? 1 : 0;
}

// Inserts a number into the stack.
void push(T_Stack *pq, int operand) {
  T_Stack nuevo = (T_Stack) malloc(sizeof(struct Node));
  if (nuevo == NULL) {
    perror("Error al insertar");
  } else {
    nuevo->number = operand;

    if (*pq == NULL) {
      nuevo->next = NULL;
    } else {
      nuevo->next = *pq;
    }
    *pq = nuevo;
  }
}

// "Inserts" an operator into the stack and operates.
// Returns true if everything OK or false in other case.
int pushOperator(T_Stack *pq, char operator) {
  int count2 = 0;
  T_Stack iter = *pq;
  while (iter != NULL && count2 < 2) {
    iter = iter->next;
    count2++;
  }

  if (count2 != 2) {
    return 0;
  } else {
    int fst, snd;
    pop(pq, &snd);
    pop(pq, &fst);
    if (operator == '+') {
      push(pq, fst + snd);
    } else if (operator == '-') {
      push(pq, fst - snd);
    } else if (operator == '*') {
      push(pq, fst * snd);
    } else if (operator == '/') {
      push(pq, fst / snd);
    } else {
```

```
60        return 0;
61      }
62      return 1;
63    }
64  }
65
66  // Puts into data the number on top of the stack, and removes the top.
67  // Returns true if everything OK or false in other case.
68  int pop(T_Stack *pq, int *data) {
69    if (*pq == NULL) {
70      return 0;
71    } else {
72      *data = (*pq)->number;
73      *pq = (*pq)->next;
74      return 1;
75    }
76  }
77
78  // Frees the memory of a stack and sets it to empty.
79  void destroy(T_Stack *pq) {
80    while(*pq != NULL) {
81      T_Stack delete = *pq;
82      *pq = (*pq)->next;
83      free(delete);
84    }
85  }
86
87
88  //TERMINA AQUI
```

```c
//EMPIEZA AQUI


#include <stdio.h>
#include <stdlib.h>
#include "Stack.h"

// Creates an empty stack.
T_Stack create() {
  return NULL;
}

// Returns true if the stack is empty and false in other case.
int isEmpty(T_Stack q) {
  return q == NULL ? 1 : 0;
}

// Inserts a number into the stack.
void push(T_Stack *pq, int operand) {
  T_Stack nuevo = (T_Stack) malloc(sizeof(struct Node));
  if (nuevo == NULL) {
    perror("Error al insertar");
  } else {
    nuevo->number = operand;

    if (*pq == NULL) {
      nuevo->next = NULL;
    } else {
      nuevo->next = *pq;
    }
    *pq = nuevo;
  }
}

// "Inserts" an operator into the stack and operates.
// Returns true if everything OK or false in other case.
int pushOperator(T_Stack *pq, char operator) {
  int count2 = 0;
  T_Stack iter = *pq;
  while (iter != NULL && count2 < 2) {
    iter = iter->next;
    count2++;
  }

  if (count2 != 2) {
    return 0;
  } else {
    int fst, snd;
    pop(pq, &snd);
    pop(pq, &fst);
    if (operator == '+') {
      push(pq, fst + snd);
    } else if (operator == '-') {
      push(pq, fst - snd);
    } else if (operator == '*') {
      push(pq, fst * snd);
    } else if (operator == '/') {
      push(pq, fst / snd);
    } else {
```

```
60        return 0;
61      }
62      return 1;
63    }
64  }
65
66  // Puts into data the number on top of the stack, and removes the top.
67  // Returns true if everything OK or false in other case.
68  int pop(T_Stack *pq, int *data) {
69    if (*pq == NULL) {
70      return 0;
71    } else {
72      *data = (*pq)->number;
73      *pq = (*pq)->next;
74      return 1;
75    }
76  }
77
78  // Frees the memory of a stack and sets it to empty.
79  void destroy(T_Stack *pq) {
80    while(*pq != NULL) {
81      T_Stack delete = *pq;
82      *pq = (*pq)->next;
83      free(delete);
84    }
85  }
86
87
88  //TERMINA AQUI
```

```c
//EMPIEZA AQUI


#include <stdio.h>
#include <stdlib.h>
#include "Stack.h"

// Creates an empty stack.
T_Stack create() {
  return NULL;
}

// Returns true if the stack is empty and false in other case.
int isEmpty(T_Stack q) {
  return q == NULL ? 1 : 0;
}

// Inserts a number into the stack.
void push(T_Stack *pq, int operand) {
  T_Stack nuevo = (T_Stack) malloc(sizeof(struct Node));
  if (nuevo == NULL) {
    perror("Error al insertar");
  } else {
    nuevo->number = operand;

    if (*pq == NULL) {
      nuevo->next = NULL;
    } else {
      nuevo->next = *pq;
    }
    *pq = nuevo;
  }
}

// "Inserts" an operator into the stack and operates.
// Returns true if everything OK or false in other case.
int pushOperator(T_Stack *pq, char operator) {
  int count2 = 0;
  T_Stack iter = *pq;
  while (iter != NULL && count2 < 2) {
    iter = iter->next;
    count2++;
  }

  if (count2 != 2) {
    return 0;
  } else {
    int fst, snd;
    pop(pq, &snd);
    pop(pq, &fst);
    if (operator == '+') {
      push(pq, fst + snd);
    } else if (operator == '-') {
      push(pq, fst - snd);
    } else if (operator == '*') {
      push(pq, fst * snd);
    } else if (operator == '/') {
      push(pq, fst / snd);
    } else {
```

```
60        return 0;
61      }
62      return 1;
63    }
64  }
65
66  // Puts into data the number on top of the stack, and removes the top.
67  // Returns true if everything OK or false in other case.
68  int pop(T_Stack *pq, int *data) {
69    if (*pq == NULL) {
70      return 0;
71    } else {
72      *data = (*pq)->number;
73      *pq = (*pq)->next;
74      return 1;
75    }
76  }
77
78  // Frees the memory of a stack and sets it to empty.
79  void destroy(T_Stack *pq) {
80    while(*pq != NULL) {
81      T_Stack delete = *pq;
82      *pq = (*pq)->next;
83      free(delete);
84    }
85  }
86
87
88  //TERMINA AQUI
```

```c
//EMPIEZA AQUI


#include <stdio.h>
#include <stdlib.h>
#include "Stack.h"

// Creates an empty stack.
T_Stack create() {
  return NULL;
}

// Returns true if the stack is empty and false in other case.
int isEmpty(T_Stack q) {
  return q == NULL ? 1 : 0;
}

// Inserts a number into the stack.
void push(T_Stack *pq, int operand) {
  T_Stack nuevo = (T_Stack) malloc(sizeof(struct Node));
  if (nuevo == NULL) {
    perror("Error al insertar");
  } else {
    nuevo->number = operand;

    if (*pq == NULL) {
      nuevo->next = NULL;
    } else {
      nuevo->next = *pq;
    }
    *pq = nuevo;
  }
}

// "Inserts" an operator into the stack and operates.
// Returns true if everything OK or false in other case.
int pushOperator(T_Stack *pq, char operator) {
  int count2 = 0;
  T_Stack iter = *pq;
  while (iter != NULL && count2 < 2) {
    iter = iter->next;
    count2++;
  }

  if (count2 != 2) {
    return 0;
  } else {
    int fst, snd;
    pop(pq, &snd);
    pop(pq, &fst);
    if (operator == '+') {
      push(pq, fst + snd);
    } else if (operator == '-') {
      push(pq, fst - snd);
    } else if (operator == '*') {
      push(pq, fst * snd);
    } else if (operator == '/') {
      push(pq, fst / snd);
    } else {
```

```
60        return 0;
61      }
62      return 1;
63    }
64  }
65
66  // Puts into data the number on top of the stack, and removes the top.
67  // Returns true if everything OK or false in other case.
68  int pop(T_Stack *pq, int *data) {
69    if (*pq == NULL) {
70      return 0;
71    } else {
72      *data = (*pq)->number;
73      *pq = (*pq)->next;
74      return 1;
75    }
76  }
77
78  // Frees the memory of a stack and sets it to empty.
79  void destroy(T_Stack *pq) {
80    while(*pq != NULL) {
81      T_Stack delete = *pq;
82      *pq = (*pq)->next;
83      free(delete);
84    }
85  }
86
87
88  //TERMINA AQUI
```

```c
//EMPIEZA AQUI


#include <stdio.h>
#include <stdlib.h>
#include "Stack.h"

// Creates an empty stack.
T_Stack create() {
  return NULL;
}

// Returns true if the stack is empty and false in other case.
int isEmpty(T_Stack q) {
  return q == NULL ? 1 : 0;
}

// Inserts a number into the stack.
void push(T_Stack *pq, int operand) {
  T_Stack nuevo = (T_Stack) malloc(sizeof(struct Node));
  if (nuevo == NULL) {
    perror("Error al insertar");
  } else {
    nuevo->number = operand;

    if (*pq == NULL) {
      nuevo->next = NULL;
    } else {
      nuevo->next = *pq;
    }
    *pq = nuevo;
  }
}

// "Inserts" an operator into the stack and operates.
// Returns true if everything OK or false in other case.
int pushOperator(T_Stack *pq, char operator) {
  int count2 = 0;
  T_Stack iter = *pq;
  while (iter != NULL && count2 < 2) {
    iter = iter->next;
    count2++;
  }

  if (count2 != 2) {
    return 0;
  } else {
    int fst, snd;
    pop(pq, &snd);
    pop(pq, &fst);
    if (operator == '+') {
      push(pq, fst + snd);
    } else if (operator == '-') {
      push(pq, fst - snd);
    } else if (operator == '*') {
      push(pq, fst * snd);
    } else if (operator == '/') {
      push(pq, fst / snd);
    } else {
```

```
60        return 0;
61      }
62      return 1;
63    }
64  }
65
66  // Puts into data the number on top of the stack, and removes the top.
67  // Returns true if everything OK or false in other case.
68  int pop(T_Stack *pq, int *data) {
69    if (*pq == NULL) {
70      return 0;
71    } else {
72      *data = (*pq)->number;
73      *pq = (*pq)->next;
74      return 1;
75    }
76  }
77
78  // Frees the memory of a stack and sets it to empty.
79  void destroy(T_Stack *pq) {
80    while(*pq != NULL) {
81      T_Stack delete = *pq;
82      *pq = (*pq)->next;
83      free(delete);
84    }
85  }
86
87
88  //TERMINA AQUI
```

```c
//EMPIEZA AQUI


#include <stdio.h>
#include <stdlib.h>
#include "Stack.h"

// Creates an empty stack.
T_Stack create() {
  return NULL;
}

// Returns true if the stack is empty and false in other case.
int isEmpty(T_Stack q) {
  return q == NULL ? 1 : 0;
}

// Inserts a number into the stack.
void push(T_Stack *pq, int operand) {
  T_Stack nuevo = (T_Stack) malloc(sizeof(struct Node));
  if (nuevo == NULL) {
    perror("Error al insertar");
  } else {
    nuevo->number = operand;

    if (*pq == NULL) {
      nuevo->next = NULL;
    } else {
      nuevo->next = *pq;
    }
    *pq = nuevo;
  }
}

// "Inserts" an operator into the stack and operates.
// Returns true if everything OK or false in other case.
int pushOperator(T_Stack *pq, char operator) {
  int count2 = 0;
  T_Stack iter = *pq;
  while (iter != NULL && count2 < 2) {
    iter = iter->next;
    count2++;
  }

  if (count2 != 2) {
    return 0;
  } else {
    int fst, snd;
    pop(pq, &snd);
    pop(pq, &fst);
    if (operator == '+') {
      push(pq, fst + snd);
    } else if (operator == '-') {
      push(pq, fst - snd);
    } else if (operator == '*') {
      push(pq, fst * snd);
    } else if (operator == '/') {
      push(pq, fst / snd);
    } else {
```

```
60        return 0;
61      }
62      return 1;
63    }
64  }
65
66  // Puts into data the number on top of the stack, and removes the top.
67  // Returns true if everything OK or false in other case.
68  int pop(T_Stack *pq, int *data) {
69    if (*pq == NULL) {
70      return 0;
71    } else {
72      *data = (*pq)->number;
73      *pq = (*pq)->next;
74      return 1;
75    }
76  }
77
78  // Frees the memory of a stack and sets it to empty.
79  void destroy(T_Stack *pq) {
80    while(*pq != NULL) {
81      T_Stack delete = *pq;
82      *pq = (*pq)->next;
83      free(delete);
84    }
85  }
86
87
88  //TERMINA AQUI
```

```c
//EMPIEZA AQUI


#include <stdio.h>
#include <stdlib.h>
#include "Stack.h"

// Creates an empty stack.
T_Stack create() {
  return NULL;
}

// Returns true if the stack is empty and false in other case.
int isEmpty(T_Stack q) {
  return q == NULL ? 1 : 0;
}

// Inserts a number into the stack.
void push(T_Stack *pq, int operand) {
  T_Stack nuevo = (T_Stack) malloc(sizeof(struct Node));
  if (nuevo == NULL) {
    perror("Error al insertar");
  } else {
    nuevo->number = operand;

    if (*pq == NULL) {
      nuevo->next = NULL;
    } else {
      nuevo->next = *pq;
    }
    *pq = nuevo;
  }
}

// "Inserts" an operator into the stack and operates.
// Returns true if everything OK or false in other case.
int pushOperator(T_Stack *pq, char operator) {
  int count2 = 0;
  T_Stack iter = *pq;
  while (iter != NULL && count2 < 2) {
    iter = iter->next;
    count2++;
  }

  if (count2 != 2) {
    return 0;
  } else {
    int fst, snd;
    pop(pq, &snd);
    pop(pq, &fst);
    if (operator == '+') {
      push(pq, fst + snd);
    } else if (operator == '-') {
      push(pq, fst - snd);
    } else if (operator == '*') {
      push(pq, fst * snd);
    } else if (operator == '/') {
      push(pq, fst / snd);
    } else {
```

```
60        return 0;
61      }
62      return 1;
63    }
64  }
65
66  // Puts into data the number on top of the stack, and removes the top.
67  // Returns true if everything OK or false in other case.
68  int pop(T_Stack *pq, int *data) {
69    if (*pq == NULL) {
70      return 0;
71    } else {
72      *data = (*pq)->number;
73      *pq = (*pq)->next;
74      return 1;
75    }
76  }
77
78  // Frees the memory of a stack and sets it to empty.
79  void destroy(T_Stack *pq) {
80    while(*pq != NULL) {
81      T_Stack delete = *pq;
82      *pq = (*pq)->next;
83      free(delete);
84    }
85  }
86
87
88  //TERMINA AQUI
```

```c
//EMPIEZA AQUI


#include <stdio.h>
#include <stdlib.h>
#include "Stack.h"

// Creates an empty stack.
T_Stack create() {
  return NULL;
}

// Returns true if the stack is empty and false in other case.
int isEmpty(T_Stack q) {
  return q == NULL ? 1 : 0;
}

// Inserts a number into the stack.
void push(T_Stack *pq, int operand) {
  T_Stack nuevo = (T_Stack) malloc(sizeof(struct Node));
  if (nuevo == NULL) {
    perror("Error al insertar");
  } else {
    nuevo->number = operand;

    if (*pq == NULL) {
      nuevo->next = NULL;
    } else {
      nuevo->next = *pq;
    }
    *pq = nuevo;
  }
}

// "Inserts" an operator into the stack and operates.
// Returns true if everything OK or false in other case.
int pushOperator(T_Stack *pq, char operator) {
  int count2 = 0;
  T_Stack iter = *pq;
  while (iter != NULL && count2 < 2) {
    iter = iter->next;
    count2++;
  }

  if (count2 != 2) {
    return 0;
  } else {
    int fst, snd;
    pop(pq, &snd);
    pop(pq, &fst);
    if (operator == '+') {
      push(pq, fst + snd);
    } else if (operator == '-') {
      push(pq, fst - snd);
    } else if (operator == '*') {
      push(pq, fst * snd);
    } else if (operator == '/') {
      push(pq, fst / snd);
    } else {
```

```
60        return 0;
61      }
62      return 1;
63    }
64  }
65
66  // Puts into data the number on top of the stack, and removes the top.
67  // Returns true if everything OK or false in other case.
68  int pop(T_Stack *pq, int *data) {
69    if (*pq == NULL) {
70      return 0;
71    } else {
72      *data = (*pq)->number;
73      *pq = (*pq)->next;
74      return 1;
75    }
76  }
77
78  // Frees the memory of a stack and sets it to empty.
79  void destroy(T_Stack *pq) {
80    while(*pq != NULL) {
81      T_Stack delete = *pq;
82      *pq = (*pq)->next;
83      free(delete);
84    }
85  }
86
87
88  //TERMINA AQUI
```

```c
//EMPIEZA AQUI


#include <stdio.h>
#include <stdlib.h>
#include "Stack.h"

// Creates an empty stack.
T_Stack create() {
  return NULL;
}

// Returns true if the stack is empty and false in other case.
int isEmpty(T_Stack q) {
  return q == NULL ? 1 : 0;
}

// Inserts a number into the stack.
void push(T_Stack *pq, int operand) {
  T_Stack nuevo = (T_Stack) malloc(sizeof(struct Node));
  if (nuevo == NULL) {
    perror("Error al insertar");
  } else {
    nuevo->number = operand;

    if (*pq == NULL) {
      nuevo->next = NULL;
    } else {
      nuevo->next = *pq;
    }
    *pq = nuevo;
  }
}

// "Inserts" an operator into the stack and operates.
// Returns true if everything OK or false in other case.
int pushOperator(T_Stack *pq, char operator) {
  int count2 = 0;
  T_Stack iter = *pq;
  while (iter != NULL && count2 < 2) {
    iter = iter->next;
    count2++;
  }

  if (count2 != 2) {
    return 0;
  } else {
    int fst, snd;
    pop(pq, &snd);
    pop(pq, &fst);
    if (operator == '+') {
      push(pq, fst + snd);
    } else if (operator == '-') {
      push(pq, fst - snd);
    } else if (operator == '*') {
      push(pq, fst * snd);
    } else if (operator == '/') {
      push(pq, fst / snd);
    } else {
```

```
60        return 0;
61      }
62      return 1;
63    }
64  }
65
66  // Puts into data the number on top of the stack, and removes the top.
67  // Returns true if everything OK or false in other case.
68  int pop(T_Stack *pq, int *data) {
69    if (*pq == NULL) {
70      return 0;
71    } else {
72      *data = (*pq)->number;
73      *pq = (*pq)->next;
74      return 1;
75    }
76  }
77
78  // Frees the memory of a stack and sets it to empty.
79  void destroy(T_Stack *pq) {
80    while(*pq != NULL) {
81      T_Stack delete = *pq;
82      *pq = (*pq)->next;
83      free(delete);
84    }
85  }
86
87
88  //TERMINA AQUI
```

```c
//EMPIEZA AQUI


#include <stdio.h>
#include <stdlib.h>
#include "Stack.h"

// Creates an empty stack.
T_Stack create() {
  return NULL;
}

// Returns true if the stack is empty and false in other case.
int isEmpty(T_Stack q) {
  return q == NULL ? 1 : 0;
}

// Inserts a number into the stack.
void push(T_Stack *pq, int operand) {
  T_Stack nuevo = (T_Stack) malloc(sizeof(struct Node));
  if (nuevo == NULL) {
    perror("Error al insertar");
  } else {
    nuevo->number = operand;

    if (*pq == NULL) {
      nuevo->next = NULL;
    } else {
      nuevo->next = *pq;
    }
    *pq = nuevo;
  }
}

// "Inserts" an operator into the stack and operates.
// Returns true if everything OK or false in other case.
int pushOperator(T_Stack *pq, char operator) {
  int count2 = 0;
  T_Stack iter = *pq;
  while (iter != NULL && count2 < 2) {
    iter = iter->next;
    count2++;
  }

  if (count2 != 2) {
    return 0;
  } else {
    int fst, snd;
    pop(pq, &snd);
    pop(pq, &fst);
    if (operator == '+') {
      push(pq, fst + snd);
    } else if (operator == '-') {
      push(pq, fst - snd);
    } else if (operator == '*') {
      push(pq, fst * snd);
    } else if (operator == '/') {
      push(pq, fst / snd);
    } else {
```

```
60        return 0;
61      }
62      return 1;
63    }
64  }
65
66  // Puts into data the number on top of the stack, and removes the top.
67  // Returns true if everything OK or false in other case.
68  int pop(T_Stack *pq, int *data) {
69    if (*pq == NULL) {
70      return 0;
71    } else {
72      *data = (*pq)->number;
73      *pq = (*pq)->next;
74      return 1;
75    }
76  }
77
78  // Frees the memory of a stack and sets it to empty.
79  void destroy(T_Stack *pq) {
80    while(*pq != NULL) {
81      T_Stack delete = *pq;
82      *pq = (*pq)->next;
83      free(delete);
84    }
85  }
86
87
88  //TERMINA AQUI
```

```c
//EMPIEZA AQUI


#include <stdio.h>
#include <stdlib.h>
#include "Stack.h"

// Creates an empty stack.
T_Stack create() {
  return NULL;
}

// Returns true if the stack is empty and false in other case.
int isEmpty(T_Stack q) {
  return q == NULL ? 1 : 0;
}

// Inserts a number into the stack.
void push(T_Stack *pq, int operand) {
  T_Stack nuevo = (T_Stack) malloc(sizeof(struct Node));
  if (nuevo == NULL) {
    perror("Error al insertar");
  } else {
    nuevo->number = operand;

    if (*pq == NULL) {
      nuevo->next = NULL;
    } else {
      nuevo->next = *pq;
    }
    *pq = nuevo;
  }
}

// "Inserts" an operator into the stack and operates.
// Returns true if everything OK or false in other case.
int pushOperator(T_Stack *pq, char operator) {
  int count2 = 0;
  T_Stack iter = *pq;
  while (iter != NULL && count2 < 2) {
    iter = iter->next;
    count2++;
  }

  if (count2 != 2) {
    return 0;
  } else {
    int fst, snd;
    pop(pq, &snd);
    pop(pq, &fst);
    if (operator == '+') {
      push(pq, fst + snd);
    } else if (operator == '-') {
      push(pq, fst - snd);
    } else if (operator == '*') {
      push(pq, fst * snd);
    } else if (operator == '/') {
      push(pq, fst / snd);
    } else {
```

```
60        return 0;
61      }
62      return 1;
63    }
64  }
65
66  // Puts into data the number on top of the stack, and removes the top.
67  // Returns true if everything OK or false in other case.
68  int pop(T_Stack *pq, int *data) {
69    if (*pq == NULL) {
70      return 0;
71    } else {
72      *data = (*pq)->number;
73      *pq = (*pq)->next;
74      return 1;
75    }
76  }
77
78  // Frees the memory of a stack and sets it to empty.
79  void destroy(T_Stack *pq) {
80    while(*pq != NULL) {
81      T_Stack delete = *pq;
82      *pq = (*pq)->next;
83      free(delete);
84    }
85  }
86
87
88  //TERMINA AQUI
```

```c
//EMPIEZA AQUI


#include <stdio.h>
#include <stdlib.h>
#include "Stack.h"

// Creates an empty stack.
T_Stack create() {
  return NULL;
}

// Returns true if the stack is empty and false in other case.
int isEmpty(T_Stack q) {
  return q == NULL ? 1 : 0;
}

// Inserts a number into the stack.
void push(T_Stack *pq, int operand) {
  T_Stack nuevo = (T_Stack) malloc(sizeof(struct Node));
  if (nuevo == NULL) {
    perror("Error al insertar");
  } else {
    nuevo->number = operand;

    if (*pq == NULL) {
      nuevo->next = NULL;
    } else {
      nuevo->next = *pq;
    }
    *pq = nuevo;
  }
}

// "Inserts" an operator into the stack and operates.
// Returns true if everything OK or false in other case.
int pushOperator(T_Stack *pq, char operator) {
  int count2 = 0;
  T_Stack iter = *pq;
  while (iter != NULL && count2 < 2) {
    iter = iter->next;
    count2++;
  }

  if (count2 != 2) {
    return 0;
  } else {
    int fst, snd;
    pop(pq, &snd);
    pop(pq, &fst);
    if (operator == '+') {
      push(pq, fst + snd);
    } else if (operator == '-') {
      push(pq, fst - snd);
    } else if (operator == '*') {
      push(pq, fst * snd);
    } else if (operator == '/') {
      push(pq, fst / snd);
    } else {
```

```
60        return 0;
61      }
62      return 1;
63    }
64  }
65
66  // Puts into data the number on top of the stack, and removes the top.
67  // Returns true if everything OK or false in other case.
68  int pop(T_Stack *pq, int *data) {
69    if (*pq == NULL) {
70      return 0;
71    } else {
72      *data = (*pq)->number;
73      *pq = (*pq)->next;
74      return 1;
75    }
76  }
77
78  // Frees the memory of a stack and sets it to empty.
79  void destroy(T_Stack *pq) {
80    while(*pq != NULL) {
81      T_Stack delete = *pq;
82      *pq = (*pq)->next;
83      free(delete);
84    }
85  }
86
87
88  //TERMINA AQUI
```

```c
//EMPIEZA AQUI


#include <stdio.h>
#include <stdlib.h>
#include "Stack.h"

// Creates an empty stack.
T_Stack create() {
  return NULL;
}

// Returns true if the stack is empty and false in other case.
int isEmpty(T_Stack q) {
  return q == NULL ? 1 : 0;
}

// Inserts a number into the stack.
void push(T_Stack *pq, int operand) {
  T_Stack nuevo = (T_Stack) malloc(sizeof(struct Node));
  if (nuevo == NULL) {
    perror("Error al insertar");
  } else {
    nuevo->number = operand;

    if (*pq == NULL) {
      nuevo->next = NULL;
    } else {
      nuevo->next = *pq;
    }
    *pq = nuevo;
  }
}

// "Inserts" an operator into the stack and operates.
// Returns true if everything OK or false in other case.
int pushOperator(T_Stack *pq, char operator) {
  int count2 = 0;
  T_Stack iter = *pq;
  while (iter != NULL && count2 < 2) {
    iter = iter->next;
    count2++;
  }

  if (count2 != 2) {
    return 0;
  } else {
    int fst, snd;
    pop(pq, &snd);
    pop(pq, &fst);
    if (operator == '+') {
      push(pq, fst + snd);
    } else if (operator == '-') {
      push(pq, fst - snd);
    } else if (operator == '*') {
      push(pq, fst * snd);
    } else if (operator == '/') {
      push(pq, fst / snd);
    } else {
```

```
60        return 0;
61      }
62      return 1;
63    }
64  }
65
66  // Puts into data the number on top of the stack, and removes the top.
67  // Returns true if everything OK or false in other case.
68  int pop(T_Stack *pq, int *data) {
69    if (*pq == NULL) {
70      return 0;
71    } else {
72      *data = (*pq)->number;
73      *pq = (*pq)->next;
74      return 1;
75    }
76  }
77
78  // Frees the memory of a stack and sets it to empty.
79  void destroy(T_Stack *pq) {
80    while(*pq != NULL) {
81      T_Stack delete = *pq;
82      *pq = (*pq)->next;
83      free(delete);
84    }
85  }
86
87
88  //TERMINA AQUI
```

```c
//EMPIEZA AQUI


#include <stdio.h>
#include <stdlib.h>
#include "Stack.h"

// Creates an empty stack.
T_Stack create() {
  return NULL;
}

// Returns true if the stack is empty and false in other case.
int isEmpty(T_Stack q) {
  return q == NULL ? 1 : 0;
}

// Inserts a number into the stack.
void push(T_Stack *pq, int operand) {
  T_Stack nuevo = (T_Stack) malloc(sizeof(struct Node));
  if (nuevo == NULL) {
    perror("Error al insertar");
  } else {
    nuevo->number = operand;

    if (*pq == NULL) {
      nuevo->next = NULL;
    } else {
      nuevo->next = *pq;
    }
    *pq = nuevo;
  }
}

// "Inserts" an operator into the stack and operates.
// Returns true if everything OK or false in other case.
int pushOperator(T_Stack *pq, char operator) {
  int count2 = 0;
  T_Stack iter = *pq;
  while (iter != NULL && count2 < 2) {
    iter = iter->next;
    count2++;
  }

  if (count2 != 2) {
    return 0;
  } else {
    int fst, snd;
    pop(pq, &snd);
    pop(pq, &fst);
    if (operator == '+') {
      push(pq, fst + snd);
    } else if (operator == '-') {
      push(pq, fst - snd);
    } else if (operator == '*') {
      push(pq, fst * snd);
    } else if (operator == '/') {
      push(pq, fst / snd);
    } else {
```

```
60        return 0;
61      }
62      return 1;
63    }
64  }
65
66  // Puts into data the number on top of the stack, and removes the top.
67  // Returns true if everything OK or false in other case.
68  int pop(T_Stack *pq, int *data) {
69    if (*pq == NULL) {
70      return 0;
71    } else {
72      *data = (*pq)->number;
73      *pq = (*pq)->next;
74      return 1;
75    }
76  }
77
78  // Frees the memory of a stack and sets it to empty.
79  void destroy(T_Stack *pq) {
80    while(*pq != NULL) {
81      T_Stack delete = *pq;
82      *pq = (*pq)->next;
83      free(delete);
84    }
85  }
86
87
88  //TERMINA AQUI
```

```c
//EMPIEZA AQUI


#include <stdio.h>
#include <stdlib.h>
#include "Stack.h"

// Creates an empty stack.
T_Stack create() {
  return NULL;
}

// Returns true if the stack is empty and false in other case.
int isEmpty(T_Stack q) {
  return q == NULL ? 1 : 0;
}

// Inserts a number into the stack.
void push(T_Stack *pq, int operand) {
  T_Stack nuevo = (T_Stack) malloc(sizeof(struct Node));
  if (nuevo == NULL) {
    perror("Error al insertar");
  } else {
    nuevo->number = operand;

    if (*pq == NULL) {
      nuevo->next = NULL;
    } else {
      nuevo->next = *pq;
    }
    *pq = nuevo;
  }
}

// "Inserts" an operator into the stack and operates.
// Returns true if everything OK or false in other case.
int pushOperator(T_Stack *pq, char operator) {
  int count2 = 0;
  T_Stack iter = *pq;
  while (iter != NULL && count2 < 2) {
    iter = iter->next;
    count2++;
  }

  if (count2 != 2) {
    return 0;
  } else {
    int fst, snd;
    pop(pq, &snd);
    pop(pq, &fst);
    if (operator == '+') {
      push(pq, fst + snd);
    } else if (operator == '-') {
      push(pq, fst - snd);
    } else if (operator == '*') {
      push(pq, fst * snd);
    } else if (operator == '/') {
      push(pq, fst / snd);
    } else {
```

```
60        return 0;
61      }
62      return 1;
63    }
64  }
65
66  // Puts into data the number on top of the stack, and removes the top.
67  // Returns true if everything OK or false in other case.
68  int pop(T_Stack *pq, int *data) {
69    if (*pq == NULL) {
70      return 0;
71    } else {
72      *data = (*pq)->number;
73      *pq = (*pq)->next;
74      return 1;
75    }
76  }
77
78  // Frees the memory of a stack and sets it to empty.
79  void destroy(T_Stack *pq) {
80    while(*pq != NULL) {
81      T_Stack delete = *pq;
82      *pq = (*pq)->next;
83      free(delete);
84    }
85  }
86
87
88  //TERMINA AQUI
```

```c
//EMPIEZA AQUI


#include <stdio.h>
#include <stdlib.h>
#include "Stack.h"

// Creates an empty stack.
T_Stack create() {
  return NULL;
}

// Returns true if the stack is empty and false in other case.
int isEmpty(T_Stack q) {
  return q == NULL ? 1 : 0;
}

// Inserts a number into the stack.
void push(T_Stack *pq, int operand) {
  T_Stack nuevo = (T_Stack) malloc(sizeof(struct Node));
  if (nuevo == NULL) {
    perror("Error al insertar");
  } else {
    nuevo->number = operand;

    if (*pq == NULL) {
      nuevo->next = NULL;
    } else {
      nuevo->next = *pq;
    }
    *pq = nuevo;
  }
}

// "Inserts" an operator into the stack and operates.
// Returns true if everything OK or false in other case.
int pushOperator(T_Stack *pq, char operator) {
  int count2 = 0;
  T_Stack iter = *pq;
  while (iter != NULL && count2 < 2) {
    iter = iter->next;
    count2++;
  }

  if (count2 != 2) {
    return 0;
  } else {
    int fst, snd;
    pop(pq, &snd);
    pop(pq, &fst);
    if (operator == '+') {
      push(pq, fst + snd);
    } else if (operator == '-') {
      push(pq, fst - snd);
    } else if (operator == '*') {
      push(pq, fst * snd);
    } else if (operator == '/') {
      push(pq, fst / snd);
    } else {
```

```
60        return 0;
61      }
62      return 1;
63    }
64  }
65
66  // Puts into data the number on top of the stack, and removes the top.
67  // Returns true if everything OK or false in other case.
68  int pop(T_Stack *pq, int *data) {
69    if (*pq == NULL) {
70      return 0;
71    } else {
72      *data = (*pq)->number;
73      *pq = (*pq)->next;
74      return 1;
75    }
76  }
77
78  // Frees the memory of a stack and sets it to empty.
79  void destroy(T_Stack *pq) {
80    while(*pq != NULL) {
81      T_Stack delete = *pq;
82      *pq = (*pq)->next;
83      free(delete);
84    }
85  }
86
87
88  //TERMINA AQUI
```

```c
//EMPIEZA AQUI


#include <stdio.h>
#include <stdlib.h>
#include "Stack.h"

// Creates an empty stack.
T_Stack create() {
  return NULL;
}

// Returns true if the stack is empty and false in other case.
int isEmpty(T_Stack q) {
  return q == NULL ? 1 : 0;
}

// Inserts a number into the stack.
void push(T_Stack *pq, int operand) {
  T_Stack nuevo = (T_Stack) malloc(sizeof(struct Node));
  if (nuevo == NULL) {
    perror("Error al insertar");
  } else {
    nuevo->number = operand;

    if (*pq == NULL) {
      nuevo->next = NULL;
    } else {
      nuevo->next = *pq;
    }
    *pq = nuevo;
  }
}

// "Inserts" an operator into the stack and operates.
// Returns true if everything OK or false in other case.
int pushOperator(T_Stack *pq, char operator) {
  int count2 = 0;
  T_Stack iter = *pq;
  while (iter != NULL && count2 < 2) {
    iter = iter->next;
    count2++;
  }

  if (count2 != 2) {
    return 0;
  } else {
    int fst, snd;
    pop(pq, &snd);
    pop(pq, &fst);
    if (operator == '+') {
      push(pq, fst + snd);
    } else if (operator == '-') {
      push(pq, fst - snd);
    } else if (operator == '*') {
      push(pq, fst * snd);
    } else if (operator == '/') {
      push(pq, fst / snd);
    } else {
```

```
60        return 0;
61      }
62      return 1;
63    }
64  }
65
66  // Puts into data the number on top of the stack, and removes the top.
67  // Returns true if everything OK or false in other case.
68  int pop(T_Stack *pq, int *data) {
69    if (*pq == NULL) {
70      return 0;
71    } else {
72      *data = (*pq)->number;
73      *pq = (*pq)->next;
74      return 1;
75    }
76  }
77
78  // Frees the memory of a stack and sets it to empty.
79  void destroy(T_Stack *pq) {
80    while(*pq != NULL) {
81      T_Stack delete = *pq;
82      *pq = (*pq)->next;
83      free(delete);
84    }
85  }
86
87
88  //TERMINA AQUI
```

```c
//EMPIEZA AQUI


#include <stdio.h>
#include <stdlib.h>
#include "Stack.h"

// Creates an empty stack.
T_Stack create() {
  return NULL;
}

// Returns true if the stack is empty and false in other case.
int isEmpty(T_Stack q) {
  return q == NULL ? 1 : 0;
}

// Inserts a number into the stack.
void push(T_Stack *pq, int operand) {
  T_Stack nuevo = (T_Stack) malloc(sizeof(struct Node));
  if (nuevo == NULL) {
    perror("Error al insertar");
  } else {
    nuevo->number = operand;

    if (*pq == NULL) {
      nuevo->next = NULL;
    } else {
      nuevo->next = *pq;
    }
    *pq = nuevo;
  }
}

// "Inserts" an operator into the stack and operates.
// Returns true if everything OK or false in other case.
int pushOperator(T_Stack *pq, char operator) {
  int count2 = 0;
  T_Stack iter = *pq;
  while (iter != NULL && count2 < 2) {
    iter = iter->next;
    count2++;
  }

  if (count2 != 2) {
    return 0;
  } else {
    int fst, snd;
    pop(pq, &snd);
    pop(pq, &fst);
    if (operator == '+') {
      push(pq, fst + snd);
    } else if (operator == '-') {
      push(pq, fst - snd);
    } else if (operator == '*') {
      push(pq, fst * snd);
    } else if (operator == '/') {
      push(pq, fst / snd);
    } else {
```

```
60        return 0;
61      }
62      return 1;
63    }
64  }
65
66  // Puts into data the number on top of the stack, and removes the top.
67  // Returns true if everything OK or false in other case.
68  int pop(T_Stack *pq, int *data) {
69    if (*pq == NULL) {
70      return 0;
71    } else {
72      *data = (*pq)->number;
73      *pq = (*pq)->next;
74      return 1;
75    }
76  }
77
78  // Frees the memory of a stack and sets it to empty.
79  void destroy(T_Stack *pq) {
80    while(*pq != NULL) {
81      T_Stack delete = *pq;
82      *pq = (*pq)->next;
83      free(delete);
84    }
85  }
86
87
88  //TERMINA AQUI
```

```c
//EMPIEZA AQUI


#include <stdio.h>
#include <stdlib.h>
#include "Stack.h"

// Creates an empty stack.
T_Stack create() {
  return NULL;
}

// Returns true if the stack is empty and false in other case.
int isEmpty(T_Stack q) {
  return q == NULL ? 1 : 0;
}

// Inserts a number into the stack.
void push(T_Stack *pq, int operand) {
  T_Stack nuevo = (T_Stack) malloc(sizeof(struct Node));
  if (nuevo == NULL) {
    perror("Error al insertar");
  } else {
    nuevo->number = operand;

    if (*pq == NULL) {
      nuevo->next = NULL;
    } else {
      nuevo->next = *pq;
    }
    *pq = nuevo;
  }
}

// "Inserts" an operator into the stack and operates.
// Returns true if everything OK or false in other case.
int pushOperator(T_Stack *pq, char operator) {
  int count2 = 0;
  T_Stack iter = *pq;
  while (iter != NULL && count2 < 2) {
    iter = iter->next;
    count2++;
  }

  if (count2 != 2) {
    return 0;
  } else {
    int fst, snd;
    pop(pq, &snd);
    pop(pq, &fst);
    if (operator == '+') {
      push(pq, fst + snd);
    } else if (operator == '-') {
      push(pq, fst - snd);
    } else if (operator == '*') {
      push(pq, fst * snd);
    } else if (operator == '/') {
      push(pq, fst / snd);
    } else {
```

```
60        return 0;
61      }
62      return 1;
63    }
64  }
65
66  // Puts into data the number on top of the stack, and removes the top.
67  // Returns true if everything OK or false in other case.
68  int pop(T_Stack *pq, int *data) {
69    if (*pq == NULL) {
70      return 0;
71    } else {
72      *data = (*pq)->number;
73      *pq = (*pq)->next;
74      return 1;
75    }
76  }
77
78  // Frees the memory of a stack and sets it to empty.
79  void destroy(T_Stack *pq) {
80    while(*pq != NULL) {
81      T_Stack delete = *pq;
82      *pq = (*pq)->next;
83      free(delete);
84    }
85  }
86
87
88  //TERMINA AQUI
```

```c
//EMPIEZA AQUI


#include <stdio.h>
#include <stdlib.h>
#include "Stack.h"

// Creates an empty stack.
T_Stack create() {
  return NULL;
}

// Returns true if the stack is empty and false in other case.
int isEmpty(T_Stack q) {
  return q == NULL ? 1 : 0;
}

// Inserts a number into the stack.
void push(T_Stack *pq, int operand) {
  T_Stack nuevo = (T_Stack) malloc(sizeof(struct Node));
  if (nuevo == NULL) {
    perror("Error al insertar");
  } else {
    nuevo->number = operand;

    if (*pq == NULL) {
      nuevo->next = NULL;
    } else {
      nuevo->next = *pq;
    }
    *pq = nuevo;
  }
}

// "Inserts" an operator into the stack and operates.
// Returns true if everything OK or false in other case.
int pushOperator(T_Stack *pq, char operator) {
  int count2 = 0;
  T_Stack iter = *pq;
  while (iter != NULL && count2 < 2) {
    iter = iter->next;
    count2++;
  }

  if (count2 != 2) {
    return 0;
  } else {
    int fst, snd;
    pop(pq, &snd);
    pop(pq, &fst);
    if (operator == '+') {
      push(pq, fst + snd);
    } else if (operator == '-') {
      push(pq, fst - snd);
    } else if (operator == '*') {
      push(pq, fst * snd);
    } else if (operator == '/') {
      push(pq, fst / snd);
    } else {
```

```
60        return 0;
61      }
62      return 1;
63    }
64  }
65
66  // Puts into data the number on top of the stack, and removes the top.
67  // Returns true if everything OK or false in other case.
68  int pop(T_Stack *pq, int *data) {
69    if (*pq == NULL) {
70      return 0;
71    } else {
72      *data = (*pq)->number;
73      *pq = (*pq)->next;
74      return 1;
75    }
76  }
77
78  // Frees the memory of a stack and sets it to empty.
79  void destroy(T_Stack *pq) {
80    while(*pq != NULL) {
81      T_Stack delete = *pq;
82      *pq = (*pq)->next;
83      free(delete);
84    }
85  }
86
87
88  //TERMINA AQUI
```

```c
//EMPIEZA AQUI


#include <stdio.h>
#include <stdlib.h>
#include "Stack.h"

// Creates an empty stack.
T_Stack create() {
  return NULL;
}

// Returns true if the stack is empty and false in other case.
int isEmpty(T_Stack q) {
  return q == NULL ? 1 : 0;
}

// Inserts a number into the stack.
void push(T_Stack *pq, int operand) {
  T_Stack nuevo = (T_Stack) malloc(sizeof(struct Node));
  if (nuevo == NULL) {
    perror("Error al insertar");
  } else {
    nuevo->number = operand;

    if (*pq == NULL) {
      nuevo->next = NULL;
    } else {
      nuevo->next = *pq;
    }
    *pq = nuevo;
  }
}

// "Inserts" an operator into the stack and operates.
// Returns true if everything OK or false in other case.
int pushOperator(T_Stack *pq, char operator) {
  int count2 = 0;
  T_Stack iter = *pq;
  while (iter != NULL && count2 < 2) {
    iter = iter->next;
    count2++;
  }

  if (count2 != 2) {
    return 0;
  } else {
    int fst, snd;
    pop(pq, &snd);
    pop(pq, &fst);
    if (operator == '+') {
      push(pq, fst + snd);
    } else if (operator == '-') {
      push(pq, fst - snd);
    } else if (operator == '*') {
      push(pq, fst * snd);
    } else if (operator == '/') {
      push(pq, fst / snd);
    } else {
```

```
60        return 0;
61      }
62      return 1;
63    }
64  }
65
66  // Puts into data the number on top of the stack, and removes the top.
67  // Returns true if everything OK or false in other case.
68  int pop(T_Stack *pq, int *data) {
69    if (*pq == NULL) {
70      return 0;
71    } else {
72      *data = (*pq)->number;
73      *pq = (*pq)->next;
74      return 1;
75    }
76  }
77
78  // Frees the memory of a stack and sets it to empty.
79  void destroy(T_Stack *pq) {
80    while(*pq != NULL) {
81      T_Stack delete = *pq;
82      *pq = (*pq)->next;
83      free(delete);
84    }
85  }
86
87
88  //TERMINA AQUI
```

```c
//EMPIEZA AQUI


#include <stdio.h>
#include <stdlib.h>
#include "Stack.h"

// Creates an empty stack.
T_Stack create() {
  return NULL;
}

// Returns true if the stack is empty and false in other case.
int isEmpty(T_Stack q) {
  return q == NULL ? 1 : 0;
}

// Inserts a number into the stack.
void push(T_Stack *pq, int operand) {
  T_Stack nuevo = (T_Stack) malloc(sizeof(struct Node));
  if (nuevo == NULL) {
    perror("Error al insertar");
  } else {
    nuevo->number = operand;

    if (*pq == NULL) {
      nuevo->next = NULL;
    } else {
      nuevo->next = *pq;
    }
    *pq = nuevo;
  }
}

// "Inserts" an operator into the stack and operates.
// Returns true if everything OK or false in other case.
int pushOperator(T_Stack *pq, char operator) {
  int count2 = 0;
  T_Stack iter = *pq;
  while (iter != NULL && count2 < 2) {
    iter = iter->next;
    count2++;
  }

  if (count2 != 2) {
    return 0;
  } else {
    int fst, snd;
    pop(pq, &snd);
    pop(pq, &fst);
    if (operator == '+') {
      push(pq, fst + snd);
    } else if (operator == '-') {
      push(pq, fst - snd);
    } else if (operator == '*') {
      push(pq, fst * snd);
    } else if (operator == '/') {
      push(pq, fst / snd);
    } else {
```

```
60        return 0;
61      }
62      return 1;
63    }
64  }
65
66  // Puts into data the number on top of the stack, and removes the top.
67  // Returns true if everything OK or false in other case.
68  int pop(T_Stack *pq, int *data) {
69    if (*pq == NULL) {
70      return 0;
71    } else {
72      *data = (*pq)->number;
73      *pq = (*pq)->next;
74      return 1;
75    }
76  }
77
78  // Frees the memory of a stack and sets it to empty.
79  void destroy(T_Stack *pq) {
80    while(*pq != NULL) {
81      T_Stack delete = *pq;
82      *pq = (*pq)->next;
83      free(delete);
84    }
85  }
86
87
88  //TERMINA AQUI
```

```c
//EMPIEZA AQUI


#include <stdio.h>
#include <stdlib.h>
#include "Stack.h"

// Creates an empty stack.
T_Stack create() {
  return NULL;
}

// Returns true if the stack is empty and false in other case.
int isEmpty(T_Stack q) {
  return q == NULL ? 1 : 0;
}

// Inserts a number into the stack.
void push(T_Stack *pq, int operand) {
  T_Stack nuevo = (T_Stack) malloc(sizeof(struct Node));
  if (nuevo == NULL) {
    perror("Error al insertar");
  } else {
    nuevo->number = operand;

    if (*pq == NULL) {
      nuevo->next = NULL;
    } else {
      nuevo->next = *pq;
    }
    *pq = nuevo;
  }
}

// "Inserts" an operator into the stack and operates.
// Returns true if everything OK or false in other case.
int pushOperator(T_Stack *pq, char operator) {
  int count2 = 0;
  T_Stack iter = *pq;
  while (iter != NULL && count2 < 2) {
    iter = iter->next;
    count2++;
  }

  if (count2 != 2) {
    return 0;
  } else {
    int fst, snd;
    pop(pq, &snd);
    pop(pq, &fst);
    if (operator == '+') {
      push(pq, fst + snd);
    } else if (operator == '-') {
      push(pq, fst - snd);
    } else if (operator == '*') {
      push(pq, fst * snd);
    } else if (operator == '/') {
      push(pq, fst / snd);
    } else {
```

```
60        return 0;
61      }
62      return 1;
63    }
64  }
65
66  // Puts into data the number on top of the stack, and removes the top.
67  // Returns true if everything OK or false in other case.
68  int pop(T_Stack *pq, int *data) {
69    if (*pq == NULL) {
70      return 0;
71    } else {
72      *data = (*pq)->number;
73      *pq = (*pq)->next;
74      return 1;
75    }
76  }
77
78  // Frees the memory of a stack and sets it to empty.
79  void destroy(T_Stack *pq) {
80    while(*pq != NULL) {
81      T_Stack delete = *pq;
82      *pq = (*pq)->next;
83      free(delete);
84    }
85  }
86
87
88  //TERMINA AQUI
```

```c
//EMPIEZA AQUI


#include <stdio.h>
#include <stdlib.h>
#include "Stack.h"

// Creates an empty stack.
T_Stack create() {
  return NULL;
}

// Returns true if the stack is empty and false in other case.
int isEmpty(T_Stack q) {
  return q == NULL ? 1 : 0;
}

// Inserts a number into the stack.
void push(T_Stack *pq, int operand) {
  T_Stack nuevo = (T_Stack) malloc(sizeof(struct Node));
  if (nuevo == NULL) {
    perror("Error al insertar");
  } else {
    nuevo->number = operand;

    if (*pq == NULL) {
      nuevo->next = NULL;
    } else {
      nuevo->next = *pq;
    }
    *pq = nuevo;
  }
}

// "Inserts" an operator into the stack and operates.
// Returns true if everything OK or false in other case.
int pushOperator(T_Stack *pq, char operator) {
  int count2 = 0;
  T_Stack iter = *pq;
  while (iter != NULL && count2 < 2) {
    iter = iter->next;
    count2++;
  }

  if (count2 != 2) {
    return 0;
  } else {
    int fst, snd;
    pop(pq, &snd);
    pop(pq, &fst);
    if (operator == '+') {
      push(pq, fst + snd);
    } else if (operator == '-') {
      push(pq, fst - snd);
    } else if (operator == '*') {
      push(pq, fst * snd);
    } else if (operator == '/') {
      push(pq, fst / snd);
    } else {
```

```
60        return 0;
61      }
62      return 1;
63    }
64  }
65
66  // Puts into data the number on top of the stack, and removes the top.
67  // Returns true if everything OK or false in other case.
68  int pop(T_Stack *pq, int *data) {
69    if (*pq == NULL) {
70      return 0;
71    } else {
72      *data = (*pq)->number;
73      *pq = (*pq)->next;
74      return 1;
75    }
76  }
77
78  // Frees the memory of a stack and sets it to empty.
79  void destroy(T_Stack *pq) {
80    while(*pq != NULL) {
81      T_Stack delete = *pq;
82      *pq = (*pq)->next;
83      free(delete);
84    }
85  }
86
87
88  //TERMINA AQUI
```

```c
//EMPIEZA AQUI


#include <stdio.h>
#include <stdlib.h>
#include "Stack.h"

// Creates an empty stack.
T_Stack create() {
  return NULL;
}

// Returns true if the stack is empty and false in other case.
int isEmpty(T_Stack q) {
  return q == NULL ? 1 : 0;
}

// Inserts a number into the stack.
void push(T_Stack *pq, int operand) {
  T_Stack nuevo = (T_Stack) malloc(sizeof(struct Node));
  if (nuevo == NULL) {
    perror("Error al insertar");
  } else {
    nuevo->number = operand;

    if (*pq == NULL) {
      nuevo->next = NULL;
    } else {
      nuevo->next = *pq;
    }
    *pq = nuevo;
  }
}

// "Inserts" an operator into the stack and operates.
// Returns true if everything OK or false in other case.
int pushOperator(T_Stack *pq, char operator) {
  int count2 = 0;
  T_Stack iter = *pq;
  while (iter != NULL && count2 < 2) {
    iter = iter->next;
    count2++;
  }

  if (count2 != 2) {
    return 0;
  } else {
    int fst, snd;
    pop(pq, &snd);
    pop(pq, &fst);
    if (operator == '+') {
      push(pq, fst + snd);
    } else if (operator == '-') {
      push(pq, fst - snd);
    } else if (operator == '*') {
      push(pq, fst * snd);
    } else if (operator == '/') {
      push(pq, fst / snd);
    } else {
```

```
60        return 0;
61      }
62      return 1;
63    }
64  }
65
66  // Puts into data the number on top of the stack, and removes the top.
67  // Returns true if everything OK or false in other case.
68  int pop(T_Stack *pq, int *data) {
69    if (*pq == NULL) {
70      return 0;
71    } else {
72      *data = (*pq)->number;
73      *pq = (*pq)->next;
74      return 1;
75    }
76  }
77
78  // Frees the memory of a stack and sets it to empty.
79  void destroy(T_Stack *pq) {
80    while(*pq != NULL) {
81      T_Stack delete = *pq;
82      *pq = (*pq)->next;
83      free(delete);
84    }
85  }
86
87
88  //TERMINA AQUI
```

```c
//EMPIEZA AQUI


#include <stdio.h>
#include <stdlib.h>
#include "Stack.h"

// Creates an empty stack.
T_Stack create() {
  return NULL;
}

// Returns true if the stack is empty and false in other case.
int isEmpty(T_Stack q) {
  return q == NULL ? 1 : 0;
}

// Inserts a number into the stack.
void push(T_Stack *pq, int operand) {
  T_Stack nuevo = (T_Stack) malloc(sizeof(struct Node));
  if (nuevo == NULL) {
    perror("Error al insertar");
  } else {
    nuevo->number = operand;

    if (*pq == NULL) {
      nuevo->next = NULL;
    } else {
      nuevo->next = *pq;
    }
    *pq = nuevo;
  }
}

// "Inserts" an operator into the stack and operates.
// Returns true if everything OK or false in other case.
int pushOperator(T_Stack *pq, char operator) {
  int count2 = 0;
  T_Stack iter = *pq;
  while (iter != NULL && count2 < 2) {
    iter = iter->next;
    count2++;
  }

  if (count2 != 2) {
    return 0;
  } else {
    int fst, snd;
    pop(pq, &snd);
    pop(pq, &fst);
    if (operator == '+') {
      push(pq, fst + snd);
    } else if (operator == '-') {
      push(pq, fst - snd);
    } else if (operator == '*') {
      push(pq, fst * snd);
    } else if (operator == '/') {
      push(pq, fst / snd);
    } else {
```

```
60        return 0;
61      }
62      return 1;
63    }
64  }
65
66  // Puts into data the number on top of the stack, and removes the top.
67  // Returns true if everything OK or false in other case.
68  int pop(T_Stack *pq, int *data) {
69    if (*pq == NULL) {
70      return 0;
71    } else {
72      *data = (*pq)->number;
73      *pq = (*pq)->next;
74      return 1;
75    }
76  }
77
78  // Frees the memory of a stack and sets it to empty.
79  void destroy(T_Stack *pq) {
80    while(*pq != NULL) {
81      T_Stack delete = *pq;
82      *pq = (*pq)->next;
83      free(delete);
84    }
85  }
86
87
88  //TERMINA AQUI
```

```c
//EMPIEZA AQUI


#include <stdio.h>
#include <stdlib.h>
#include "Stack.h"

// Creates an empty stack.
T_Stack create() {
  return NULL;
}

// Returns true if the stack is empty and false in other case.
int isEmpty(T_Stack q) {
  return q == NULL ? 1 : 0;
}

// Inserts a number into the stack.
void push(T_Stack *pq, int operand) {
  T_Stack nuevo = (T_Stack) malloc(sizeof(struct Node));
  if (nuevo == NULL) {
    perror("Error al insertar");
  } else {
    nuevo->number = operand;

    if (*pq == NULL) {
      nuevo->next = NULL;
    } else {
      nuevo->next = *pq;
    }
    *pq = nuevo;
  }
}

// "Inserts" an operator into the stack and operates.
// Returns true if everything OK or false in other case.
int pushOperator(T_Stack *pq, char operator) {
  int count2 = 0;
  T_Stack iter = *pq;
  while (iter != NULL && count2 < 2) {
    iter = iter->next;
    count2++;
  }

  if (count2 != 2) {
    return 0;
  } else {
    int fst, snd;
    pop(pq, &snd);
    pop(pq, &fst);
    if (operator == '+') {
      push(pq, fst + snd);
    } else if (operator == '-') {
      push(pq, fst - snd);
    } else if (operator == '*') {
      push(pq, fst * snd);
    } else if (operator == '/') {
      push(pq, fst / snd);
    } else {
```

```
60        return 0;
61      }
62      return 1;
63    }
64  }
65
66  // Puts into data the number on top of the stack, and removes the top.
67  // Returns true if everything OK or false in other case.
68  int pop(T_Stack *pq, int *data) {
69    if (*pq == NULL) {
70      return 0;
71    } else {
72      *data = (*pq)->number;
73      *pq = (*pq)->next;
74      return 1;
75    }
76  }
77
78  // Frees the memory of a stack and sets it to empty.
79  void destroy(T_Stack *pq) {
80    while(*pq != NULL) {
81      T_Stack delete = *pq;
82      *pq = (*pq)->next;
83      free(delete);
84    }
85  }
86
87
88  //TERMINA AQUI
```

```c
//EMPIEZA AQUI


#include <stdio.h>
#include <stdlib.h>
#include "Stack.h"

// Creates an empty stack.
T_Stack create() {
  return NULL;
}

// Returns true if the stack is empty and false in other case.
int isEmpty(T_Stack q) {
  return q == NULL ? 1 : 0;
}

// Inserts a number into the stack.
void push(T_Stack *pq, int operand) {
  T_Stack nuevo = (T_Stack) malloc(sizeof(struct Node));
  if (nuevo == NULL) {
    perror("Error al insertar");
  } else {
    nuevo->number = operand;

    if (*pq == NULL) {
      nuevo->next = NULL;
    } else {
      nuevo->next = *pq;
    }
    *pq = nuevo;
  }
}

// "Inserts" an operator into the stack and operates.
// Returns true if everything OK or false in other case.
int pushOperator(T_Stack *pq, char operator) {
  int count2 = 0;
  T_Stack iter = *pq;
  while (iter != NULL && count2 < 2) {
    iter = iter->next;
    count2++;
  }

  if (count2 != 2) {
    return 0;
  } else {
    int fst, snd;
    pop(pq, &snd);
    pop(pq, &fst);
    if (operator == '+') {
      push(pq, fst + snd);
    } else if (operator == '-') {
      push(pq, fst - snd);
    } else if (operator == '*') {
      push(pq, fst * snd);
    } else if (operator == '/') {
      push(pq, fst / snd);
    } else {
```

```
60        return 0;
61      }
62      return 1;
63    }
64  }
65
66  // Puts into data the number on top of the stack, and removes the top.
67  // Returns true if everything OK or false in other case.
68  int pop(T_Stack *pq, int *data) {
69    if (*pq == NULL) {
70      return 0;
71    } else {
72      *data = (*pq)->number;
73      *pq = (*pq)->next;
74      return 1;
75    }
76  }
77
78  // Frees the memory of a stack and sets it to empty.
79  void destroy(T_Stack *pq) {
80    while(*pq != NULL) {
81      T_Stack delete = *pq;
82      *pq = (*pq)->next;
83      free(delete);
84    }
85  }
86
87
88  //TERMINA AQUI
```

```c
//EMPIEZA AQUI


#include <stdio.h>
#include <stdlib.h>
#include "Stack.h"

// Creates an empty stack.
T_Stack create() {
  return NULL;
}

// Returns true if the stack is empty and false in other case.
int isEmpty(T_Stack q) {
  return q == NULL ? 1 : 0;
}

// Inserts a number into the stack.
void push(T_Stack *pq, int operand) {
  T_Stack nuevo = (T_Stack) malloc(sizeof(struct Node));
  if (nuevo == NULL) {
    perror("Error al insertar");
  } else {
    nuevo->number = operand;

    if (*pq == NULL) {
      nuevo->next = NULL;
    } else {
      nuevo->next = *pq;
    }
    *pq = nuevo;
  }
}

// "Inserts" an operator into the stack and operates.
// Returns true if everything OK or false in other case.
int pushOperator(T_Stack *pq, char operator) {
  int count2 = 0;
  T_Stack iter = *pq;
  while (iter != NULL && count2 < 2) {
    iter = iter->next;
    count2++;
  }

  if (count2 != 2) {
    return 0;
  } else {
    int fst, snd;
    pop(pq, &snd);
    pop(pq, &fst);
    if (operator == '+') {
      push(pq, fst + snd);
    } else if (operator == '-') {
      push(pq, fst - snd);
    } else if (operator == '*') {
      push(pq, fst * snd);
    } else if (operator == '/') {
      push(pq, fst / snd);
    } else {
```

```
60        return 0;
61      }
62      return 1;
63    }
64  }
65
66  // Puts into data the number on top of the stack, and removes the top.
67  // Returns true if everything OK or false in other case.
68  int pop(T_Stack *pq, int *data) {
69    if (*pq == NULL) {
70      return 0;
71    } else {
72      *data = (*pq)->number;
73      *pq = (*pq)->next;
74      return 1;
75    }
76  }
77
78  // Frees the memory of a stack and sets it to empty.
79  void destroy(T_Stack *pq) {
80    while(*pq != NULL) {
81      T_Stack delete = *pq;
82      *pq = (*pq)->next;
83      free(delete);
84    }
85  }
86
87
88  //TERMINA AQUI
```

```c
//EMPIEZA AQUI


#include <stdio.h>
#include <stdlib.h>
#include "Stack.h"

// Creates an empty stack.
T_Stack create() {
  return NULL;
}

// Returns true if the stack is empty and false in other case.
int isEmpty(T_Stack q) {
  return q == NULL ? 1 : 0;
}

// Inserts a number into the stack.
void push(T_Stack *pq, int operand) {
  T_Stack nuevo = (T_Stack) malloc(sizeof(struct Node));
  if (nuevo == NULL) {
    perror("Error al insertar");
  } else {
    nuevo->number = operand;

    if (*pq == NULL) {
      nuevo->next = NULL;
    } else {
      nuevo->next = *pq;
    }
    *pq = nuevo;
  }
}

// "Inserts" an operator into the stack and operates.
// Returns true if everything OK or false in other case.
int pushOperator(T_Stack *pq, char operator) {
  int count2 = 0;
  T_Stack iter = *pq;
  while (iter != NULL && count2 < 2) {
    iter = iter->next;
    count2++;
  }

  if (count2 != 2) {
    return 0;
  } else {
    int fst, snd;
    pop(pq, &snd);
    pop(pq, &fst);
    if (operator == '+') {
      push(pq, fst + snd);
    } else if (operator == '-') {
      push(pq, fst - snd);
    } else if (operator == '*') {
      push(pq, fst * snd);
    } else if (operator == '/') {
      push(pq, fst / snd);
    } else {
```

```
60        return 0;
61      }
62      return 1;
63    }
64  }
65
66  // Puts into data the number on top of the stack, and removes the top.
67  // Returns true if everything OK or false in other case.
68  int pop(T_Stack *pq, int *data) {
69    if (*pq == NULL) {
70      return 0;
71    } else {
72      *data = (*pq)->number;
73      *pq = (*pq)->next;
74      return 1;
75    }
76  }
77
78  // Frees the memory of a stack and sets it to empty.
79  void destroy(T_Stack *pq) {
80    while(*pq != NULL) {
81      T_Stack delete = *pq;
82      *pq = (*pq)->next;
83      free(delete);
84    }
85  }
86
87
88  //TERMINA AQUI
```

```c
//EMPIEZA AQUI


#include <stdio.h>
#include <stdlib.h>
#include "Stack.h"

// Creates an empty stack.
T_Stack create() {
  return NULL;
}

// Returns true if the stack is empty and false in other case.
int isEmpty(T_Stack q) {
  return q == NULL ? 1 : 0;
}

// Inserts a number into the stack.
void push(T_Stack *pq, int operand) {
  T_Stack nuevo = (T_Stack) malloc(sizeof(struct Node));
  if (nuevo == NULL) {
    perror("Error al insertar");
  } else {
    nuevo->number = operand;

    if (*pq == NULL) {
      nuevo->next = NULL;
    } else {
      nuevo->next = *pq;
    }
    *pq = nuevo;
  }
}

// "Inserts" an operator into the stack and operates.
// Returns true if everything OK or false in other case.
int pushOperator(T_Stack *pq, char operator) {
  int count2 = 0;
  T_Stack iter = *pq;
  while (iter != NULL && count2 < 2) {
    iter = iter->next;
    count2++;
  }

  if (count2 != 2) {
    return 0;
  } else {
    int fst, snd;
    pop(pq, &snd);
    pop(pq, &fst);
    if (operator == '+') {
      push(pq, fst + snd);
    } else if (operator == '-') {
      push(pq, fst - snd);
    } else if (operator == '*') {
      push(pq, fst * snd);
    } else if (operator == '/') {
      push(pq, fst / snd);
    } else {
```

```
60        return 0;
61      }
62      return 1;
63    }
64  }
65
66  // Puts into data the number on top of the stack, and removes the top.
67  // Returns true if everything OK or false in other case.
68  int pop(T_Stack *pq, int *data) {
69    if (*pq == NULL) {
70      return 0;
71    } else {
72      *data = (*pq)->number;
73      *pq = (*pq)->next;
74      return 1;
75    }
76  }
77
78  // Frees the memory of a stack and sets it to empty.
79  void destroy(T_Stack *pq) {
80    while(*pq != NULL) {
81      T_Stack delete = *pq;
82      *pq = (*pq)->next;
83      free(delete);
84    }
85  }
86
87
88  //TERMINA AQUI
```

```c
//EMPIEZA AQUI


#include <stdio.h>
#include <stdlib.h>
#include "Stack.h"

// Creates an empty stack.
T_Stack create() {
  return NULL;
}

// Returns true if the stack is empty and false in other case.
int isEmpty(T_Stack q) {
  return q == NULL ? 1 : 0;
}

// Inserts a number into the stack.
void push(T_Stack *pq, int operand) {
  T_Stack nuevo = (T_Stack) malloc(sizeof(struct Node));
  if (nuevo == NULL) {
    perror("Error al insertar");
  } else {
    nuevo->number = operand;

    if (*pq == NULL) {
      nuevo->next = NULL;
    } else {
      nuevo->next = *pq;
    }
    *pq = nuevo;
  }
}

// "Inserts" an operator into the stack and operates.
// Returns true if everything OK or false in other case.
int pushOperator(T_Stack *pq, char operator) {
  int count2 = 0;
  T_Stack iter = *pq;
  while (iter != NULL && count2 < 2) {
    iter = iter->next;
    count2++;
  }

  if (count2 != 2) {
    return 0;
  } else {
    int fst, snd;
    pop(pq, &snd);
    pop(pq, &fst);
    if (operator == '+') {
      push(pq, fst + snd);
    } else if (operator == '-') {
      push(pq, fst - snd);
    } else if (operator == '*') {
      push(pq, fst * snd);
    } else if (operator == '/') {
      push(pq, fst / snd);
    } else {
```

```
60        return 0;
61      }
62      return 1;
63    }
64  }
65
66  // Puts into data the number on top of the stack, and removes the top.
67  // Returns true if everything OK or false in other case.
68  int pop(T_Stack *pq, int *data) {
69    if (*pq == NULL) {
70      return 0;
71    } else {
72      *data = (*pq)->number;
73      *pq = (*pq)->next;
74      return 1;
75    }
76  }
77
78  // Frees the memory of a stack and sets it to empty.
79  void destroy(T_Stack *pq) {
80    while(*pq != NULL) {
81      T_Stack delete = *pq;
82      *pq = (*pq)->next;
83      free(delete);
84    }
85  }
86
87
88  //TERMINA AQUI
```

```c
//EMPIEZA AQUI


#include <stdio.h>
#include <stdlib.h>
#include "Stack.h"

// Creates an empty stack.
T_Stack create() {
  return NULL;
}

// Returns true if the stack is empty and false in other case.
int isEmpty(T_Stack q) {
  return q == NULL ? 1 : 0;
}

// Inserts a number into the stack.
void push(T_Stack *pq, int operand) {
  T_Stack nuevo = (T_Stack) malloc(sizeof(struct Node));
  if (nuevo == NULL) {
    perror("Error al insertar");
  } else {
    nuevo->number = operand;

    if (*pq == NULL) {
      nuevo->next = NULL;
    } else {
      nuevo->next = *pq;
    }
    *pq = nuevo;
  }
}

// "Inserts" an operator into the stack and operates.
// Returns true if everything OK or false in other case.
int pushOperator(T_Stack *pq, char operator) {
  int count2 = 0;
  T_Stack iter = *pq;
  while (iter != NULL && count2 < 2) {
    iter = iter->next;
    count2++;
  }

  if (count2 != 2) {
    return 0;
  } else {
    int fst, snd;
    pop(pq, &snd);
    pop(pq, &fst);
    if (operator == '+') {
      push(pq, fst + snd);
    } else if (operator == '-') {
      push(pq, fst - snd);
    } else if (operator == '*') {
      push(pq, fst * snd);
    } else if (operator == '/') {
      push(pq, fst / snd);
    } else {
```

```
60        return 0;
61      }
62      return 1;
63    }
64  }
65
66  // Puts into data the number on top of the stack, and removes the top.
67  // Returns true if everything OK or false in other case.
68  int pop(T_Stack *pq, int *data) {
69    if (*pq == NULL) {
70      return 0;
71    } else {
72      *data = (*pq)->number;
73      *pq = (*pq)->next;
74      return 1;
75    }
76  }
77
78  // Frees the memory of a stack and sets it to empty.
79  void destroy(T_Stack *pq) {
80    while(*pq != NULL) {
81      T_Stack delete = *pq;
82      *pq = (*pq)->next;
83      free(delete);
84    }
85  }
86
87
88  //TERMINA AQUI
```

```c
//EMPIEZA AQUI


#include <stdio.h>
#include <stdlib.h>
#include "Stack.h"

// Creates an empty stack.
T_Stack create() {
  return NULL;
}

// Returns true if the stack is empty and false in other case.
int isEmpty(T_Stack q) {
  return q == NULL ? 1 : 0;
}

// Inserts a number into the stack.
void push(T_Stack *pq, int operand) {
  T_Stack nuevo = (T_Stack) malloc(sizeof(struct Node));
  if (nuevo == NULL) {
    perror("Error al insertar");
  } else {
    nuevo->number = operand;

    if (*pq == NULL) {
      nuevo->next = NULL;
    } else {
      nuevo->next = *pq;
    }
    *pq = nuevo;
  }
}

// "Inserts" an operator into the stack and operates.
// Returns true if everything OK or false in other case.
int pushOperator(T_Stack *pq, char operator) {
  int count2 = 0;
  T_Stack iter = *pq;
  while (iter != NULL && count2 < 2) {
    iter = iter->next;
    count2++;
  }

  if (count2 != 2) {
    return 0;
  } else {
    int fst, snd;
    pop(pq, &snd);
    pop(pq, &fst);
    if (operator == '+') {
      push(pq, fst + snd);
    } else if (operator == '-') {
      push(pq, fst - snd);
    } else if (operator == '*') {
      push(pq, fst * snd);
    } else if (operator == '/') {
      push(pq, fst / snd);
    } else {
```

```
60        return 0;
61      }
62      return 1;
63    }
64  }
65
66  // Puts into data the number on top of the stack, and removes the top.
67  // Returns true if everything OK or false in other case.
68  int pop(T_Stack *pq, int *data) {
69    if (*pq == NULL) {
70      return 0;
71    } else {
72      *data = (*pq)->number;
73      *pq = (*pq)->next;
74      return 1;
75    }
76  }
77
78  // Frees the memory of a stack and sets it to empty.
79  void destroy(T_Stack *pq) {
80    while(*pq != NULL) {
81      T_Stack delete = *pq;
82      *pq = (*pq)->next;
83      free(delete);
84    }
85  }
86
87
88  //TERMINA AQUI
```

```c
//EMPIEZA AQUI


#include <stdio.h>
#include <stdlib.h>
#include "Stack.h"

// Creates an empty stack.
T_Stack create() {
  return NULL;
}

// Returns true if the stack is empty and false in other case.
int isEmpty(T_Stack q) {
  return q == NULL ? 1 : 0;
}

// Inserts a number into the stack.
void push(T_Stack *pq, int operand) {
  T_Stack nuevo = (T_Stack) malloc(sizeof(struct Node));
  if (nuevo == NULL) {
    perror("Error al insertar");
  } else {
    nuevo->number = operand;

    if (*pq == NULL) {
      nuevo->next = NULL;
    } else {
      nuevo->next = *pq;
    }
    *pq = nuevo;
  }
}

// "Inserts" an operator into the stack and operates.
// Returns true if everything OK or false in other case.
int pushOperator(T_Stack *pq, char operator) {
  int count2 = 0;
  T_Stack iter = *pq;
  while (iter != NULL && count2 < 2) {
    iter = iter->next;
    count2++;
  }

  if (count2 != 2) {
    return 0;
  } else {
    int fst, snd;
    pop(pq, &snd);
    pop(pq, &fst);
    if (operator == '+') {
      push(pq, fst + snd);
    } else if (operator == '-') {
      push(pq, fst - snd);
    } else if (operator == '*') {
      push(pq, fst * snd);
    } else if (operator == '/') {
      push(pq, fst / snd);
    } else {
```

```
60        return 0;
61      }
62      return 1;
63    }
64  }
65
66  // Puts into data the number on top of the stack, and removes the top.
67  // Returns true if everything OK or false in other case.
68  int pop(T_Stack *pq, int *data) {
69    if (*pq == NULL) {
70      return 0;
71    } else {
72      *data = (*pq)->number;
73      *pq = (*pq)->next;
74      return 1;
75    }
76  }
77
78  // Frees the memory of a stack and sets it to empty.
79  void destroy(T_Stack *pq) {
80    while(*pq != NULL) {
81      T_Stack delete = *pq;
82      *pq = (*pq)->next;
83      free(delete);
84    }
85  }
86
87
88  //TERMINA AQUI
```

```c
//EMPIEZA AQUI


#include <stdio.h>
#include <stdlib.h>
#include "Stack.h"

// Creates an empty stack.
T_Stack create() {
  return NULL;
}

// Returns true if the stack is empty and false in other case.
int isEmpty(T_Stack q) {
  return q == NULL ? 1 : 0;
}

// Inserts a number into the stack.
void push(T_Stack *pq, int operand) {
  T_Stack nuevo = (T_Stack) malloc(sizeof(struct Node));
  if (nuevo == NULL) {
    perror("Error al insertar");
  } else {
    nuevo->number = operand;

    if (*pq == NULL) {
      nuevo->next = NULL;
    } else {
      nuevo->next = *pq;
    }
    *pq = nuevo;
  }
}

// "Inserts" an operator into the stack and operates.
// Returns true if everything OK or false in other case.
int pushOperator(T_Stack *pq, char operator) {
  int count2 = 0;
  T_Stack iter = *pq;
  while (iter != NULL && count2 < 2) {
    iter = iter->next;
    count2++;
  }

  if (count2 != 2) {
    return 0;
  } else {
    int fst, snd;
    pop(pq, &snd);
    pop(pq, &fst);
    if (operator == '+') {
      push(pq, fst + snd);
    } else if (operator == '-') {
      push(pq, fst - snd);
    } else if (operator == '*') {
      push(pq, fst * snd);
    } else if (operator == '/') {
      push(pq, fst / snd);
    } else {
```

```
60        return 0;
61      }
62      return 1;
63    }
64  }
65
66  // Puts into data the number on top of the stack, and removes the top.
67  // Returns true if everything OK or false in other case.
68  int pop(T_Stack *pq, int *data) {
69    if (*pq == NULL) {
70      return 0;
71    } else {
72      *data = (*pq)->number;
73      *pq = (*pq)->next;
74      return 1;
75    }
76  }
77
78  // Frees the memory of a stack and sets it to empty.
79  void destroy(T_Stack *pq) {
80    while(*pq != NULL) {
81      T_Stack delete = *pq;
82      *pq = (*pq)->next;
83      free(delete);
84    }
85  }
86
87
88  //TERMINA AQUI
```

```c
//EMPIEZA AQUI


#include <stdio.h>
#include <stdlib.h>
#include "Stack.h"

// Creates an empty stack.
T_Stack create() {
  return NULL;
}

// Returns true if the stack is empty and false in other case.
int isEmpty(T_Stack q) {
  return q == NULL ? 1 : 0;
}

// Inserts a number into the stack.
void push(T_Stack *pq, int operand) {
  T_Stack nuevo = (T_Stack) malloc(sizeof(struct Node));
  if (nuevo == NULL) {
    perror("Error al insertar");
  } else {
    nuevo->number = operand;

    if (*pq == NULL) {
      nuevo->next = NULL;
    } else {
      nuevo->next = *pq;
    }
    *pq = nuevo;
  }
}

// "Inserts" an operator into the stack and operates.
// Returns true if everything OK or false in other case.
int pushOperator(T_Stack *pq, char operator) {
  int count2 = 0;
  T_Stack iter = *pq;
  while (iter != NULL && count2 < 2) {
    iter = iter->next;
    count2++;
  }

  if (count2 != 2) {
    return 0;
  } else {
    int fst, snd;
    pop(pq, &snd);
    pop(pq, &fst);
    if (operator == '+') {
      push(pq, fst + snd);
    } else if (operator == '-') {
      push(pq, fst - snd);
    } else if (operator == '*') {
      push(pq, fst * snd);
    } else if (operator == '/') {
      push(pq, fst / snd);
    } else {
```

```
60        return 0;
61      }
62      return 1;
63    }
64  }
65
66  // Puts into data the number on top of the stack, and removes the top.
67  // Returns true if everything OK or false in other case.
68  int pop(T_Stack *pq, int *data) {
69    if (*pq == NULL) {
70      return 0;
71    } else {
72      *data = (*pq)->number;
73      *pq = (*pq)->next;
74      return 1;
75    }
76  }
77
78  // Frees the memory of a stack and sets it to empty.
79  void destroy(T_Stack *pq) {
80    while(*pq != NULL) {
81      T_Stack delete = *pq;
82      *pq = (*pq)->next;
83      free(delete);
84    }
85  }
86
87
88  //TERMINA AQUI
```