```c

#include <stdio.h>
#include <stdlib.h>
#include "../include/stk.h"

void printError(char* pString)
{
  printf("ERROR: %s\n", pString);
}

void printSuccess(char* pString)
{
  printf("SUCCESS: %s\n", pString);
}

/*****************************************************************************
 Function:    main

 Description: test all the functionality of the stack

 Parameters:  none
 Returned:    none

 *****************************************************************************/

int main()
{
  Stack sTheStack;

  DATATYPE values[3] = {65,66,67};
  const int VALUES_LEN = 3;
  int i;
  DATATYPE data;

  stkCreate(&sTheStack);

  if( stkIsFull(sTheStack))
  {
    printError("The stack should not be full");
    /*return -1;*/
  }
  else
  {
    printSuccess("The stack should is not full");
  }
  if( !stkIsEmpty(sTheStack))
  {
```

```
48    printError("The stack should be empty");
49    /*return -1;*/
50  }
51  else
52  {
53    printSuccess("The stack is empty");
54  }
55
56
57  // create stack
58
59  // TEST POP
60
61  // the stack is empty but created
62  if( stkPop(&sTheStack, &data) != ERROR_STACK_EMPTY )
63  {
64    printf("FAILURE: Pop did not return error on empty stack\n");
65  }
66  else
67  {
68    printf("SUCCESS: Pop returned ERROR_STACK_EMPTY on empty stack\n");
69  }
70
71  // the stack is NULL
72  if( stkPop(NULL, &data) != ERROR_NO_STACK_CREATE )
73  {
74    printf ("FAILURE: Pop did not return error on NULL stack\n");
75  }
76  else
77  {
78    printf ("SUCCESS: Pop returned ERROR_NO_STACK_CREATE on NULL stack\n");
79  }
80
81  // the data value is NULL
82  if( stkPop(&sTheStack, NULL) != ERROR_NO_STACK_CREATE )
83  {
84    printf ("FAILURE: Pop did not return error on NULL data value\n");
85  }
86  else
87  {
88    printf ("SUCCESS: Pop returned ERROR_NO_STACK_CREATE ");
89    printf ("on NULL data value\n");
90  }
91
92  // tamper with top
93  sTheStack.top = -9;
94  if( stkPop(&sTheStack, &data) != ERROR_NO_STACK_CREATE )
95  {
```

```
 96    printf("FAILURE: Pop did not return error on stack with ");
 97    printf("invalid top (-9)\n");
 98  }
 99  else
100  {
101    printf("SUCCESS: Pop returned ERROR_NO_STACK_CREATE on stack with ");
102    printf("invalid top (-9)\n");
103  }
104
105  sTheStack.top = MAX_STACK * 2 ;
106  if( stkPop(&sTheStack, &data) != ERROR_NO_STACK_CREATE )
107  {
108    printf("FAILURE: Pop did not return error on stack with ");
109    printf("invalid top (%d)\n",MAX_STACK * 2);
110  }
111  else
112  {
113    printf("SUCCESS: Pop returned ERROR_NO_STACK_CREATE on stack with ");
114    printf("invalid top (%d)\n",MAX_STACK * 2);
115  }
116
117  // reset stack
118  stkTerminate(&sTheStack);
119  stkCreate(&sTheStack);
120
121
122  // the stack is already created and is valid and empty
123  // add some data.  make sure push works
124  for(i=0; i<  VALUES_LEN; i++)
125  {
126    if( stkPush(&sTheStack, values[i]) != NO_ERROR)
127    {
128      printf("FAILURE: Push of %c failed\n", values[i]);
129    }
130    else
131    {
132      printf("SUCCESS: Push of %c\n", values[i]);
133    }
134  }
135
136  // pop values off the stack, check to see that we get the
137  // correct values
138  for(i=0; i< VALUES_LEN; i++)
139  {
140    if( stkPop(&sTheStack, &data) != NO_ERROR )
141    {
142      printf("FAILURE: Pop failed\n");
143    }
```

```
144     else
145     {
146       if( data == values[VALUES_LEN-1-i])
147       {
148         printf("SUCCESS: Pop of %c\n", data);
149       }
150       else
151       {
152         printf("FAILURE: Pop received %c expected %c\n",
153             data, values[VALUES_LEN-1-i]);
154       }
155 
156     }
157   }
158 
159   // make sure if we pop off an emptied stack we still get the correct
160   // errors
161   for(i=0; i< VALUES_LEN; i++)
162   {
163     if( stkPop(&sTheStack, &data) != ERROR_STACK_EMPTY )
164     {
165       printf("FAILURE: Pop failed to return ERROR_STACK_EMPTY ");
166       printf("on empty stack\n");
167     }
168     else
169     {
170       printf("SUCCESS: Pop returned ERROR_STACK_EMPTY on empty stack\n");
171     }
172   }
173 
174   stkTerminate(&sTheStack);
175 
176 
177   return EXIT_SUCCESS;
178 
179 }
180 
```