```cpp
/**
 * @file -  LinkedListStackMain.cpp
 *
 * Uses LinkedListStack<T> that has implemented StackADT<T> using a linked list as the internal data structure
 * for the stack elements.
 *
 * @author - Humayun Kabir, Instructor, CSCI 161, VIU
 * @version - 0.0.1
 * @date - April 25, 2021
 *
 */




#include <iostream>
#include <string>
#include "StackADT.h"
#include "LinkedListStack.cpp"

using namespace std;

LinkedListStack<string> getColorStack() {
    int count = 7;
    string colors[] = {"Red", "Orange", "Yellow", "Green", "Cyan", "Blue", "Violet"};
    LinkedListStack<string> linkedliststack;
    for (int i = 0; i<count; i++) {
        linkedliststack.push(colors[i]);
    }
    return linkedliststack;
}

int main(){


  cout << "************** Testing LinkedListStack ***************" << endl;

  /*
   * LinkedListStack default constructor
   */
  LinkedListStack<double> linkedliststackdouble1;

  cout << "LinkedListStack<double> default constructor..." << endl;
  cout << "LinkedListStack<double> isEmpty: " << linkedliststackdouble1.isEmpty() << endl;
  cout << "LinkedListStack<double> size: " << linkedliststackdouble1.getSize() << endl;

  try {
  cout << "LinkedListStack<double> pushing 10.01"<<endl;
  linkedliststackdouble1.push(10.01);
  cout << "LinkedListStack<double> peek(): " << linkedliststackdouble1.peek() << endl;

  cout << "LinkedListStack<double> pushing 20.02"<<endl;
  linkedliststackdouble1.push(20.02);
  cout << "LinkedListStack<double> peek(): " << linkedliststackdouble1.peek() << endl;

  cout << "LinkedListStack<double> pushing 30.03"<<endl;
  linkedliststackdouble1.push(30.03);
  cout << "LinkedListStack<double> peek(): " << linkedliststackdouble1.peek() << endl;

  cout << "LinkedListStack<double> pushing 40.04"<<endl;
  linkedliststackdouble1.push(40.04);
  cout << "LinkedListStack<double> peek(): " << linkedliststackdouble1.peek() << endl;

  cout << "LinkedListStack<double> pushing 50.05"<<endl;
  linkedliststackdouble1.push(50.05);
  cout << "LinkedListStack<double> peek(): " << linkedliststackdouble1.peek() << endl;

```

```
68      }
69      catch (const char* excp) {
70          cout<<excp<<endl;
71      }
72      cout <<"LinkedListStack<double> after pushing 10.01, 20.02, 30.03, 40.04, and 50.05" <<
         endl;
73      cout << "LinkedListStack<double> isEmpty: " << linkedliststackdouble1.isEmpty() << endl
        ;
74      cout << "LinkedListStack<double> size: " << linkedliststackdouble1.getSize() << endl;
75
76      try {
77        cout << "LinkedListStack<double> pop(): " << linkedliststackdouble1.pop() << endl;
78        cout << "LinkedListStack<double> pop(): " << linkedliststackdouble1.pop() << endl;
79        cout << "LinkedListStack<double> pop(): " << linkedliststackdouble1.pop() << endl;
80        cout << "LinkedListStack<double> pop(): " << linkedliststackdouble1.pop() << endl;
81        cout << "LinkedListStack<double> pop(): " << linkedliststackdouble1.pop() << endl;
82        cout << "LinkedListStack<double> pop(): " << linkedliststackdouble1.pop() << endl;
83      }
84      catch (const char* excp) {
85          cout << excp << endl;
86      }
87
88      cout << "LinkedListStack<double> isEmpty: " << linkedliststackdouble1.isEmpty() << endl
        ;
89      cout << "LinkedListStack<double> size: " << linkedliststackdouble1.getSize() << endl;
90
91      try {
92        cout << "LinkedListStack<double> pushing 1.1"<<endl;
93        linkedliststackdouble1.push(1.1);
94        cout << "LinkedListStack<double> peek(): " << linkedliststackdouble1.peek() << endl;
95        cout << "LinkedListStack<double> pushing 2.2"<<endl;
96        linkedliststackdouble1.push(2.2);
97        cout << "LinkedListStack<double> peek(): " << linkedliststackdouble1.peek() << endl;
98        cout << "LinkedListStack<double> pushing 3.3"<<endl;
99        linkedliststackdouble1.push(3.3);
100       cout << "LinkedListStack<double> peek(): " << linkedliststackdouble1.peek() << endl;
101       cout << "LinkedListStack<double> pushing 4.4"<<endl;
102       linkedliststackdouble1.push(4.4);
103       cout << "LinkedListStack<double> peek(): " << linkedliststackdouble1.peek() << endl;
104       cout << "LinkedListStack<double> pushing 5.5"<<endl;
105       linkedliststackdouble1.push(5.5);
106       cout << "LinkedListStack<double> peek(): " << linkedliststackdouble1.peek() << endl;
107       cout << "LinkedListStack<double> pushing 6.6"<<endl;
108       linkedliststackdouble1.push(6.6);
109       cout << "LinkedListStack<double> peek(): " << linkedliststackdouble1.peek() << endl;
110       cout << "LinkedListStack<double> pushing 7.7"<<endl;
111       linkedliststackdouble1.push(7.7);
112       cout << "LinkedListStack<double> peek(): " << linkedliststackdouble1.peek() << endl;
113       cout << "LinkedListStack<double> pushing 8.8"<<endl;
114       linkedliststackdouble1.push(8.8);
115       cout << "LinkedListStack<double> peek(): " << linkedliststackdouble1.peek() << endl;
116       cout << "LinkedListStack<double> pushing 9.9"<<endl;
117       linkedliststackdouble1.push(9.9);
118       cout << "LinkedListStack<double> peek(): " << linkedliststackdouble1.peek() << endl;
119       cout << "LinkedListStack<double> pushing 10.10"<<endl;
120       linkedliststackdouble1.push(10.10);
121       cout << "LinkedListStack<double> peek(): " << linkedliststackdouble1.peek() << endl;
122
123
124     }
125     catch (const char* excp) {
126         cout << excp << endl;
127     }
128
129     cout << "LinkedListStack<double> isEmpty: " << linkedliststackdouble1.isEmpty() << endl
        ;
130     cout << "LinkedListStack<double> size: " << linkedliststackdouble1.getSize() << endl;
131
132
```

```
133     /*
134      * LinkedListStack copy constructor
135      */
136     cout << "LinkedListStack<double> copy  constructor..." << endl;
137     LinkedListStack<double> linkedliststackdouble3(linkedliststackdouble1);
138 
139     cout << "LinkedListStack<double> isEmpty: " << linkedliststackdouble3.isEmpty() << endl
        ;
140     cout << "LinkedListStack<double> size: " << linkedliststackdouble3.getSize() << endl;
141 
142     while(linkedliststackdouble3.getSize() > 0) {
143         try {
144           cout << "LinkedListStack<double> pop(): "<< linkedliststackdouble3.pop() << endl;
145         }
146         catch (const char* excp) {
147             cout << excp << endl;
148         }
149     }
150 
151     cout << "LinkedListStack<double> isEmpty: " << linkedliststackdouble3.isEmpty() << endl
        ;
152     cout << "LinkedListStack<double> size: " << linkedliststackdouble3.getSize() << endl;
153 
154 
155 
156     /*
157      * LinkedListStack move constructor.
158      */
159     cout << "LinkedListStack<string> move constructor......" <<endl;
160     LinkedListStack<string> linkedliststackstring1 = getColorStack();
161 
162     cout << "LinkedListStack<string> isEmpty: " << linkedliststackstring1.isEmpty() << endl
        ;
163     cout << "LinkedListStack<string> size: " << linkedliststackstring1.getSize() << endl;
164 
165 
166     while(linkedliststackstring1.getSize() > 0) {
167         try {
168           cout << "LinkedListStack<string> pop(): "<< linkedliststackstring1.pop() << endl;
169         }
170         catch (const char* excp) {
171             cout << excp << endl;
172         }
173     }
174 
175 
176     cout << "LinkedListStack<string> isEmpty: " << linkedliststackstring1.isEmpty() << endl
        ;
177     cout << "LinkedListStack<string> size: " << linkedliststackstring1.getSize() << endl;
178 
179 
180     /*
181      * LinkedListStack copy assignment
182      */
183     cout << "LinkedListStack<double> copy assignment..." << endl;
184     LinkedListStack<double> linkedliststackdouble4;
185 
186     cout << "LinkedListStack<double> before copy assignment" << endl;
187     cout << "LinkedListStack<double> isEmpty: " << linkedliststackdouble4.isEmpty() << endl
        ;
188     cout << "LinkedListStack<double> size: " << linkedliststackdouble4.getSize() << endl;
189 
190     linkedliststackdouble4 = linkedliststackdouble1;
191 
192 
193     cout << "LinkedListStack<double> after copy assignment" << endl;
194     cout << "LinkedListStack<double> isEmpty: " << linkedliststackdouble4.isEmpty() << endl
        ;
195     cout << "LinkedListStack<double> size: " << linkedliststackdouble4.getSize() << endl;
```

```
196
197     while(linkedliststackdouble4.getSize() > 0) {
198         try {
199           cout << "LinkedListStack<double> pop(): "<< linkedliststackdouble4.pop() << endl;
200         }
201         catch (const char* excp) {
202             cout << excp << endl;
203         }
204     }
205
206     cout << "LinkedListStack<double> isEmpty: " << linkedliststackdouble4.isEmpty() << endl
        ;
207     cout << "LinkedListStack<double> size: " << linkedliststackdouble4.getSize() << endl;
208
209
210
211      /*
212      * LinkedListStack move assignment.
213      */
214     cout << "LinkedListStack<string> move assignment......" <<endl;
215     LinkedListStack<string> linkedliststackstring2;
216
217     cout << "LinkedListStack<string> before move assignment" << endl;
218     cout << "LinkedListStack<string> isEmpty: " << linkedliststackstring2.isEmpty() << endl
        ;
219     cout << "LinkedListStack<string> size: " << linkedliststackstring2.getSize() << endl;
220
221
222     linkedliststackstring2 = getColorStack();
223
224     cout << "LinkedListStack<string> after move assignment" << endl;
225     cout << "LinkedListStack<string> isEmpty: " << linkedliststackstring2.isEmpty() << endl
        ;
226     cout << "LinkedListStack<string> size: " << linkedliststackstring2.getSize() << endl;
227
228
229     while(linkedliststackstring2.getSize() > 0) {
230         try {
231           cout << "LinkedListStack<string> pop(): "<< linkedliststackstring2.pop() << endl;
232         }
233         catch (const char* excp) {
234             cout << excp << endl;
235         }
236     }
237
238
239     cout << "LinkedListStack<string> isEmpty: " << linkedliststackstring2.isEmpty() << endl
        ;
240     cout << "LinkedListStack<string> size: " << linkedliststackstring2.getSize() << endl;
241
242
243     cout << "LinkedListStack<int>*
        ............................................................"<< endl;
244     LinkedListStack<int>* linkedliststack = NULL;
245
246     linkedliststack = new LinkedListStack<int>();
247
248     cout << "LinkedListStack<int>* isEmpty: " << linkedliststack->isEmpty() << endl;
249     cout << "LinkedListStack<int>* size: " << linkedliststack->getSize() << endl;
250
251     try{
252       linkedliststack->push(100);
253
254       cout << "LinkedListStack<int>* isEmpty after pushing 100: " << linkedliststack->
          isEmpty() << endl;
255       cout << "LinkedListStack<int>* size after pushing 100: " << linkedliststack->getSize
          () << endl;
256       cout << "LinkedListStack<int>* top after pushing 100: " << linkedliststack->peek() <<
           endl;
```

```
257
258       linkedliststack->push(200);
259
260       cout << "LinkedListStack<int>* isEmpty after pushing 200: " << linkedliststack->
          isEmpty() << endl;
261       cout << "LinkedListStack<int>* size after pushing 200: " << linkedliststack->getSize
          () << endl;
262       cout << "LinkedListStack<int>* top after pushing 200: " << linkedliststack->peek() <<
           endl;
263
264       linkedliststack->push(300);
265
266       cout << "LinkedListStack<int>* isEmpty after pushing 300: " << linkedliststack->
          isEmpty() << endl;
267       cout << "LinkedListStack<int>* size after pushing 300: " << linkedliststack->getSize
          () << endl;
268       cout << "LinkedListStack<int>* top after pushing 300: " << linkedliststack->peek() <<
           endl;
269
270       int popped = linkedliststack->pop();
271
272       cout << "LinkedListStack<int>* isEmpty after popping: " << popped << ": " <<
          linkedliststack->isEmpty() << endl;
273       cout << "LinkedListStack<int>* size after popping: " << popped << ": " <<
          linkedliststack->getSize() << endl;
274       cout << "LinkedListStack<int>* top after popping: " << popped << ": " <<
          linkedliststack->peek() << endl;
275
276       linkedliststack->push(300);
277
278       int peeked = linkedliststack->peek();
279
280       cout << "LinkedListStack<int>* isEmpty after peeking: " << peeked << ": " <<
          linkedliststack->isEmpty() << endl;
281       cout << "LinkedListStack<int>* size after peeking: " << peeked << ": " <<
          linkedliststack->getSize() << endl;
282       cout << "LinkedListStack<int>* top after peeking: " << peeked << ": " <<
          linkedliststack->peek() << endl;
283
284       linkedliststack->push(400);
285       linkedliststack->push(500);
286       linkedliststack->push(600);
287       linkedliststack->push(700);
288       linkedliststack->push(800);
289       linkedliststack->push(900);
290       linkedliststack->push(1000);
291
292       cout << "LinkedListStack<int>* isEmpty after pushing 1000: " << linkedliststack->
          isEmpty() << endl;
293       cout << "LinkedListStack<int>* size after pushing 1000: " << linkedliststack->getSize
          () << endl;
294       cout << "LinkedListStack<int>* top after pushing 1000: " << linkedliststack->peek()
          << endl;
295
296       linkedliststack->push(1100);
297       cout << "LinkedListStack<int>* isEmpty after pushing 1100: " << linkedliststack->
          isEmpty() << endl;
298       cout << "LinkedListStack<int>* size after pushing 1100: " << linkedliststack->getSize
          () << endl;
299       cout << "LinkedListStack<int>* top after pushing 1100: " << linkedliststack->peek()
          << endl;
300     }
301     catch (const char* excp){
302         cerr << excp << endl;
303     }
304     int stackSize = linkedliststack->getSize();
305     try{
306       for (int i=0; i<stackSize; i++) {
307         cout << "LinkedListStack<int>* pop(): " << linkedliststack->pop() << endl;
```

```
308       }
309       cout << "ArrayStck<int>* pop() " << linkedliststack->pop() << endl;
310     }
311     catch (const char* excp){
312         cerr << excp << endl;
313     }
314 
315     cout << "StackADT<int>* ................................................." << endl;
316     StackADT<int>* stack = NULL;
317 
318     stack = new LinkedListStack<int>();
319 
320     cout << "StackADT<int>* isEmpty: " << stack->isEmpty() << endl;
321     cout << "StackADT<int>* size: " << stack->getSize() << endl;
322 
323     try{
324       stack->push(100);
325 
326       cout << "StackADT<int>* isEmpty after pushing 100: " << stack->isEmpty() << endl;
327       cout << "StackADT<int>* size after pushing 100: " << stack->getSize() << endl;
328       cout << "StackADT<int>* top after pushing 100: " << stack->peek() << endl;
329 
330       stack->push(200);
331 
332       cout << "StackADT<int>* isEmpty after pushing 200: " << stack->isEmpty() << endl;
333       cout << "StackADT<int>* size after pushing 200: " << stack->getSize() << endl;
334       cout << "StackADT<int>* top after pushing 200: " << stack->peek() << endl;
335 
336       stack->push(300);
337 
338       cout << "StackADT<int>* isEmpty after pushing 300: " << stack->isEmpty() << endl;
339       cout << "StackADT<int>* size after pushing 300: " << stack->getSize() << endl;
340       cout << "StackADT<int>* top after pushing 300: " << stack->peek() << endl;
341 
342       int popped = stack->pop();
343 
344       cout << "StackADT<int>* isEmpty after popping: " << popped << ": " << stack->isEmpty
          () << endl;
345       cout << "StackADT<int>* size after popping: " << popped << ": " << stack->getSize()
          << endl;
346       cout << "StackADT<int>* top after popping: " << popped << ": " << stack->peek() <<
          endl;
347 
348       stack->push(300);
349 
350       int peeked = stack->peek();
351 
352       cout << "StackADT<int>* isEmpty after peeking: " << peeked << ": " << stack->isEmpty
          () << endl;
353       cout << "StackADT<int>* size after peeking: " << peeked << ": " << stack->getSize()
          << endl;
354       cout << "StackADT<int>* top after peeking: " << peeked << ": " << stack->peek() <<
          endl;
355 
356       stack->push(400);
357       stack->push(500);
358       stack->push(600);
359       stack->push(700);
360       stack->push(800);
361       stack->push(900);
362       stack->push(1000);
363 
364       cout << "StackADT<int>* isEmpty after pushing 1000: " << stack->isEmpty() << endl;
365       cout << "StackADT<int>* size after pushing 1000: " << stack->getSize() << endl;
366       cout << "StackADT<int>* top after pushing 1000: " << stack->peek() << endl;
367 
368       stack->push(1100);
369       cout << "StackADT<int>* isEmpty after pushing 1100: " << stack->isEmpty() << endl;
370       cout << "StackADT<int>* size after pushing 1100: " << stack->getSize() << endl;
```

```
371       cout << "StackADT<int>* top after pushing 1100: " << stack->peek() << endl;
372     }
373     catch (const char* excp){
374         cerr << excp << endl;
375     }
376     stackSize = stack->getSize();
377     try{
378       for (int i=0; i<stackSize; i++) {
379         cout << "StackADT<int> pop(): " << stack->pop() << endl;
380       }
381       cout << "StackADT<int> pop(): " << stack->pop() << endl;
382     }
383     catch (const char* excp){
384         cerr << excp << endl;
385     }
386 
387     /*
388      * Releasing memory from dynimically coreated objects
389      */
390     if(linkedliststack != NULL) {
391       delete linkedliststack;
392     }
393 
394     if(stack != NULL) {
395         delete stack;
396     }
397 
398     cout << "************** End of Testing LinkedListStack ***************" << endl;
399 
400 
401     return 0;
402   }
403 
```