

```

1  #include "SList.h"
2
3  /***** constructor/destructor definitions *****/
4
5  SList::SList()
6  : head( NULL ), size( 0 )
7  {
8      /* empty */
9  }
10
11 SList::~~SList()
12 {
13     this->clear();
14 }
15
16 /***** public function definitions *****/
17
18 void SList::insertHead( int headValue )
19 {
20     if( size > 0 )
21     {
22         SLNode *tempValue  = this->head;
23
24         this->head = new SLNode( headValue );
25         size++;
26         head->setNextNode( tempValue );
27     }
28     else
29     {
30         this->head = new SLNode( headValue );
31         size++;
32     }
33 }
34
35 void SList::insertTail( int tailValue )
36 {
37     if( size > 0 )
38     {
39         SLNode *tempValue  = head;
40
41         while( tempValue->getNextNode() != NULL )
42         {
43             tempValue = tempValue->getNextNode();
44         }
45
46         tempValue->setNextNode( new SLNode( tailValue ) );
47     }
48     else
49     {
50         head = new SLNode( tailValue );

```

```

51     }
52
53     size++;
54 }
55
56 void SList::removeHead()
57 {
58     if( size > 0 )
59     {
60         SLNode *tempValue = (this->head)->getNextNode();
61
62         delete this->head;
63         this->head = tempValue;
64         size--;
65     }
66 }
67
68 void SList::removeTail()
69 {
70     if( size > 1 )
71     {
72         SLNode *tempValue = head;
73
74         while( (tempValue->getNextNode())->getNextNode() != NULL )
75         {
76             tempValue = tempValue->getNextNode();
77         }
78
79         delete tempValue->getNextNode();
80         tempValue->setNextNode( NULL );
81         size--;
82     }
83     else if( size == 1 )
84     {
85         delete this->head;
86         size--;
87     }
88 }
89
90 void SList::clear()
91 {
92     if( size > 1 )
93     {
94         SLNode *tempValue;
95
96         do {
97             tempValue = (this->head)->getNextNode();
98             delete this->head;
99             this->head = tempValue;
100

```

```

101         } while( (this->head)->getNextNode() != NULL );
102     }
103
104     if( size >= 1 )
105     {
106         delete this->head;
107         size = 0;
108     }
109
110 }
111
112 string SList::toString() const
113 {
114
115     if( size > 0 )
116     {
117         stringstream ss;
118
119         if( (this->head)->getNextNode() == NULL )
120         {
121             ss << (this->head)->getContents();
122             return ss.str();
123         }
124
125         SLNode *tempValue = head;
126
127         do {
128             ss << tempValue->getContents() << ",";
129             tempValue = tempValue->getNextNode();
130
131         } while( tempValue->getNextNode() != NULL );
132
133         ss << tempValue->getContents();
134
135         return ss.str();
136     }
137     else
138     {
139         return "";
140     }
141 }
142
143 /***** accessor/mutator function definitions *****/
144 unsigned int SList::getSize() const
145 {
146     return this->size;
147 }
148

```