```c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

/* Solution to Problem 2, part 1: a single type that could be used for
representing integers or strings. */

typedef union{
 int num;
 char * str;
}StrorIntType;

/* Solution to Problem 2, part 2: a type that could be used for representing
either integer or string binary search trees. */
typedef enum {integer, string} kind;

struct node{
 StrorIntType elem;
 kind k;
 struct node *left, *right;
};

struct node *create(StrorIntType val, kind k) {
 struct node *new =  (struct node *)malloc(sizeof(struct node));
 new->elem = val;
 new->k = k;
 new->left = NULL;
 new->right = NULL;
 return new;
}

/* Solution to Problem 2, part 3: a function for determining if a given integer
or string appears in a given binary search tree*/

int isEqual(StrorIntType elem1, StrorIntType elem2, kind k) {
 if (k == 0) {
  if (elem1.num == elem2.num) {
   return 1;
  }
 }
 else if (k == 1) {
  if (strcmp(elem1.str, elem2.str) == 0) {
   return 1;
  }
 }
 return 0;

}

int compare(StrorIntType elem1, StrorIntType elem2, kind k) {
 if (k == 0) {
  return elem1.num - elem2.num;
 } else {
```

```
54    return strcmp(elem1.str, elem2.str);
55   }
56  }
57
58  int member(struct node* node, StrorIntType val,
59   int (*eq)(StrorIntType, StrorIntType, kind),
60   int (*ord)(StrorIntType, StrorIntType, kind)) {
61
62    if (node == NULL ) {
63     return 0;
64    }
65    if ((*eq)(node->elem, val, node->k) == 1) {
66     return 1;
67    }
68
69    if ((*ord)(node->elem, val, node->k) < 0) {
70     return member(node->right, val, isEqual, compare);
71    }
72
73    return member(node->left, val, isEqual, compare);
74  }
75
76  /* Solution to Problem 2, part 4: a function for inserting a given integer or
77  string into a given binary search tree */
78
79  struct node* insert(struct node* node, StrorIntType val,
80   int (*ord)(StrorIntType, StrorIntType, kind), kind k) {
81   if (node == NULL) {
82    return create(val, k);
83   }
84
85   /* left Subtree is unchanged */
86   if ((*ord)(node->elem, val, k) < 0) {
87    node->right  = insert(node->right, val, compare, k);
88   }
89   /* Right Subtree is unchanged */
90   else if ((*ord)(node->elem, val, k) > 0) {
91    node->left = insert(node->left, val, compare, k);
92   }
93
94   return node;
95  }
96
97  /* Solution to Problem 1, part 5: a function to print elements in an integer
98  or string binary search tree using an inorder traversal*/
99
100 void printVal(StrorIntType val, kind k)  {
101  if (k == 0) {
102   printf("%d\n", val.num);
103  } else {
104   printf("%s\n", val.str);
105  }
106 }
107
108 void printtree(struct node *node, void (*prt)(StrorIntType, kind k)) {
109  if (node != NULL) {
```

```
110
111   printtree(node->left, printVal);
112
113   (*prt)(node->elem, node->k);
114
115   printtree(node->right, printVal);
116  }
117 }
118
119 void memberTest(int res) {
120  if (res == 1) {
121   printf("is a member of the tree\n");
122  } else {
123   printf("is not a member of the tree\n");
124  }
125 }
126
127 int main() {
128  struct node *tree = NULL;
129
130  /* Test 1 For Integer tree */
131  printf("Test 1: Integer tree\n");
132  StrorIntType value;
133  value.num = 10;
134  struct node *intTree = insert(tree, value, compare, integer);
135  value.num = 3;
136  insert(intTree, value, compare, integer);
137  value.num = 22;
138  insert(intTree, value, compare, integer);
139  value.num = 15;
140  insert(intTree, value, compare, integer);
141  value.num = 9;
142  insert(intTree, value, compare, integer);
143  printf("Tree:\n");
144  printtree(intTree, printVal);
145
146  value.num = 10;
147  int res = member(intTree, value, isEqual, compare);
148  printf("%d ", value.num);
149  memberTest(res);
150
151  value.num = 12;
152  res = member(intTree, value, isEqual, compare);
153  printf("%d ", value.num);
154  memberTest(res);
155
156  /* Test 2 For String tree */
157  printf("\nTest 2: String tree\n");
158  StrorIntType value2;
159  value2.str = "ardvark";
160  struct node *strTree = insert(tree, value2, compare, string);
161  value2.str = "goose";
162  insert(strTree, value2, compare, string);
163  value2.str = "beetle";
164  insert(strTree, value2, compare, string);
165  value2.str = "zebra";
```

```
166   insert(strTree, value2, compare, string);
167   value2.str = "eagle";
168   insert(strTree, value2, compare, string);
169   printf("Tree:\n");
170   printtree(strTree, printVal);
171
172   value2.str = "ardvark";
173   res = member(strTree, value2, isEqual, compare);
174   printf("%s ", value2.str);
175   memberTest(res);
176
177   value2.str = "butterfly";
178   res = member(strTree, value2, isEqual, compare);
179   printf("%s ", value2.str);
180   memberTest(res);
181
182   return 0;
183 }
```

PDF document made with CodePrint using Prism