```c
/* RBTreeTest.c */
#include <stdio.h>
#include <string.h>
#include <malloc.h>
#include <stdlib.h>
#include "RBTree.h"

static int IntCmp(const void *keyAddr, const void *dataAddr)
{
    int *p1 = (int *)keyAddr;
    int *p2 = (int *)dataAddr;
    return (*p1 - *p2);
}

static void IntTraverse(void *dataAddr)
{
    int *p = (int *)dataAddr;
    printf("%d\n", *p);
}

static int StringCmp(const void *keyAddr, const void *dataAddr)
{
    char *p1 = *(char **)keyAddr;
    char *p2 = *(char **)dataAddr;
    return strcmp(p1, p2);
}

static void StringTraverse(void *dataAddr)
{
    char *p = *(char **)dataAddr;
    printf("%s\n", p);
}

static void StringFree(void *dataAddr)
{
    char *p = *(char **)dataAddr;
    free(p);
}

int main()
{
    RBTREE intRBTree;
    RBTreeNew(&intRBTree, sizeof(int), IntCmp, NULL);
    int i = 0;
    for (; i < 10; i++)
    {
        RBTreeInsert(&intRBTree, &i);
    }
    if (!RBTreeEmpty(&intRBTree))
    {
        printf("intRBTree size is %d\n", RBTreeSize(&intRBTree));
        printf("intRBTree height is %d\n", RBTreeHeight(&intRBTree));
        RBTreeTravIn(&intRBTree, IntTraverse);
    }
    int intRemove = 1;
    if (0 == RBTreeRemove(&intRBTree, &intRemove))
    {
        printf("intRBTree remove key %d success\n", intRemove);
    }
    else
    {
        printf("intRBTree remove key %d fail\n", intRemove);
    }
    if (0 == RBTreeRemove(&intRBTree, &intRemove))
    {
        printf("intRBTree remove key %d success\n", intRemove);
    }
    else
    {
        printf("intRBTree remove key %d fail\n", intRemove);
    }
    if (!RBTreeEmpty(&intRBTree))
    {
```

```
74          printf("intRBTree size is %d\n", RBTreeSize(&intRBTree));
75          printf("intRBTree height is %d\n", RBTreeHeight(&intRBTree));
76          RBTreeTravInRec(&intRBTree, IntTraverse);
77      }
78
79      int intSearch = 2;
80      RBTREENODE *node = RBTreeSearch(&intRBTree, &intSearch);
81      if (NULL != node)
82      {
83          printf("key %d is in intRBTree\n", intSearch);
84      }
85      else
86      {
87          printf("key %d is not in intRBTree\n", intSearch);
88      }
89      intSearch = 11;
90      node = RBTreeSearch(&intRBTree, &intSearch);
91      if (NULL != node)
92      {
93          printf("key %d is in intRBTree\n", intSearch);
94      }
95      else
96      {
97          printf("key %d is not in intRBTree\n", intSearch);
98      }
99      RBTreeDispose(&intRBTree);
100
101     printf("\n\n");
102
103     RBTREE stringRBTree;
104     RBTreeNew(&stringRBTree, sizeof(char *), StringCmp, StringFree);
105     char *name1 = strdup("pc");
106     char *name2 = strdup("pcwl513");
107     char *name3 = strdup("pcpc");
108     char *name4 = strdup("jerry");
109     char *name5 = strdup("jerry.peng");
110     char *name6 = strdup("yanglupu");
111     char *name7 = strdup("zhanglei");
112     char *name8 = strdup("lishanke");
113     RBTreeInsert(&stringRBTree, &name1);
114     RBTreeInsert(&stringRBTree, &name2);
115     RBTreeInsert(&stringRBTree, &name3);
116     RBTreeInsert(&stringRBTree, &name4);
117     RBTreeInsert(&stringRBTree, &name5);
118     RBTreeInsert(&stringRBTree, &name6);
119     RBTreeInsert(&stringRBTree, &name7);
120     RBTreeInsert(&stringRBTree, &name8);
121     if (!RBTreeEmpty(&stringRBTree))
122     {
123         printf("stringRBTree size is %d\n", RBTreeSize(&stringRBTree));
124         printf("stringRBTree height is %d\n", RBTreeHeight(&stringRBTree));
125         RBTreeTravIn(&stringRBTree, StringTraverse);
126     }
127
128     char *strRemove = "pcpc";
129     if (0 == RBTreeRemove(&stringRBTree, &strRemove))
130     {
131         printf("stringRBTree remove key %s success\n", strRemove);
132     }
133     else
134     {
135         printf("stringRBTree remove key %s fail\n", strRemove);
136     }
137     if (0 == RBTreeRemove(&stringRBTree, &strRemove))
138     {
139         printf("stringRBTree remove key %s success\n", strRemove);
140     }
141     else
142     {
143         printf("stringRBTree remove key %s fail\n", strRemove);
144     }
145     if (!RBTreeEmpty(&stringRBTree))
146     {
```

```
147             printf("stringRBTree size is %d\n", RBTreeSize(&stringRBTree));
148             printf("stringRBTree height is %d\n", RBTreeHeight(&stringRBTree));
149             RBTreeTravInRec(&stringRBTree, StringTraverse);
150         }
151 
152         char *strSearch = "yanglupu";
153         node = RBTreeSearch(&stringRBTree, &strSearch);
154         if (NULL != node)
155         {
156             printf("key %s is in stringRBTree\n", strSearch);
157         }
158         else
159         {
160             printf("key %s is not in stringRBTree\n", strSearch);
161         }
162         strSearch = "123";
163         node = RBTreeSearch(&stringRBTree, &strSearch);
164         if (NULL != node)
165         {
166             printf("key %s is in stringRBTree\n", strSearch);
167         }
168         else
169         {
170             printf("key %s is not in stringRBTree\n", strSearch);
171         }
172         RBTreeDispose(&stringRBTree);
173         return 0;
174     }
```