encodeChar

Write a function that accepts two character strings ***s*** and ***t***, an array of structures and the size of array as parameters, encodes the characters in ***s*** to ***t***, and passes the encoded string ***t*** to the caller via call by reference. During the encoding process, each ***source*** character is converted into the corresponding ***code*** character based on the user defined rules. For other source characters, the `code` will be the same as the `source`. In addition, write a function `createTable()` that accepts two parameters, `table` and `size`. It allows users to define encoding rules in `table`. For example, when the following rules `'a'→'d'; 'b'→'z'; 'z'→'a'; 'd'→'b'` are defined with `size = 4`, if the character string **s** is `"abort"`, then the encoded string ***t*** will be `"dzort"`. The structure `Rule` is defined below:

```
typedef struct {
   char source;
   char code;
} Rule;
```

The function prototypes are given below:

```
void createTable(Rule *table, int *size);
void encodeChar(Rule *table, int size, char *s, char *t);
```

A sample program template is given below to test the function:

```
#include <stdio.h>
#include <string.h>
typedef struct {
   char source;
   char code;
} Rule;
void createTable(Rule *table, int *size);
void printTable(Rule *table, int size);
void encodeChar(Rule *table, int size, char *s, char *t);
int main()
{
   char s[80], t[80], dummychar, *p;
   int size, choice;
   Rule table[100];

   printf("Select one of the following options:\n");
   printf("1: createTable()\n");
   printf("2: printTable()\n");
   printf("3: encodeChar()\n");
   printf("4: exit()\n");
   do {
      printf("Enter your choice: \n");
      scanf("%d", &choice);
      switch (choice) {
         case 1:
            printf("createTable(): \n");
            createTable(table, &size);
            break;
         case 2:
```

```
            printf("printTable(): \n");
            printTable(table, size);
            break;
         case 3:
            scanf("%c",&dummychar);
            printf("Source string: \n");
            fgets(s, 80, stdin);
            if (p=strchr(s,'\n')) *p = '\0';
            encodeChar(table,size,s,t);
            printf("Encoded string: %s\n", t);
            break;
         default:
            break;
      }
   } while (choice < 4);
   return 0;
}
void printTable(Rule *table, int size)
{
   int i;

   for (i=0; i<size; i++)
   {
      printf("%d: %c->%c\n", i+1, table->source, table->code);
      table++;
   }
}
void createTable(Rule *table, int *size)
{
   /* Write your code here */
}
void encodeChar(Rule *table, int size, char *s, char *t)
{
   /* Write your code here */
}
```

Some sample input and output sessions are given below:

(1) Test Case 1:

```
Select one of the following options:
1: createTable()
2: printTable()
3: encodeChar()
4: exit()
Enter your choice:
1
createTable():
Enter number of rules:
3
Enter rule 1:
Enter source character:
a
Enter code character:
```

```
b
Enter rule 2:
Enter source character:
b
Enter code character:
c
Enter rule 3:
Enter source character:
c
Enter code character:
d
Enter your choice:
2
printTable():
1: a->b
2: b->c
3: c->d
Enter your choice:
4
```

(2) Test Case 2:

```
<use Test Case 1>
Enter your choice:
3
Source string:
abcd
Encoded string: bcdd
Enter your choice:
Enter your choice:
4
```

(3) Test Case 3:

```
<use Test Case 1>
Enter your choice:
3
Source string:
abort abort
Encoded string: bcort bcort
Enter your choice:
4
```

(4) Test Case 4:

```
Select one of the following options:
1: createTable()
2: printTable()
3: encodeChar()
4: exit()
Enter your choice:
1
createTable():
Enter number of rules:
2
Enter rule 1:
Enter source character:
```

```
a
Enter code character:
d
Enter rule 2:
Enter source character:
t
Enter code character:
b
Enter your choice:
3
Source string:
abort
Encoded string: dborb
Enter your choice:
4
```