WEEK 7 LAB - RECURSION

Questions 1-3

You may use the program template in Figure 1 to test your recursive functions developed in this lab. The program contains a `main()` which includes a switch statement so that the following functions can be tested by the user. Write the code for each function and use the suggested test cases to test your code for correctness.

```
#include <stdio.h>

/* function prototypes */
int rNumDigits1(int num);
void rNumDigits2(int num, int *result);
int rDigitPos1(int num, int digit);
void rDigitPos2(int num, int digit, int *pos);
int rSquare1(int num);
void rSquare2(int num, int *result);

int main()
{
   int choice;
   int number;
   int digit, result=0;

   do {
      printf("\nPerform the following functions ITERATIVELY:\n");
      printf("1:  rNumDigits1()\n");
      printf("2:  rNumDigits2()\n");
      printf("3:  rDigitPos1()\n");
      printf("4:  rDigitPos2()\n");
      printf("5:  rSquare1()\n");
      printf("6:  rSquare2()\n");
      printf("7:  quit\n");
      printf("Enter your choice: ");
      scanf("%d", &choice);

      switch (choice) {
         case 1:
            printf("Enter the number: \n");
            scanf("%d", &number);
            printf("rNumDigits1(): %d\n", rNumDigits1(number));
            break;
         case 2:
            printf("Enter the number: \n");
            scanf("%d", &number);
            rNumDigits2(number, &result);
            printf("rNumDigits2(): %d\n", result);
            break;
         case 3:
            printf("Enter the number: \n");
            scanf("%d", &number);
            printf("Enter the digit: \n");
            scanf("%d", &digit);
            printf("rDigitPos1(): %d\n", rDigitPos1(number,
digit));
            break;
         case 4:
            printf("Enter the number: \n");
            scanf("%d", &number);
            printf("Enter the digit: \n");
            scanf("%d", &digit);
```

```
             rDigitPos2(number, digit, &result);
             printf("rDigitPos2(): %d\n", result);
             break;
          case 5:
             printf("Enter the number: \n");
             scanf("%d", &number);
             printf("rSquare1(): %d\n", rSquare1(number));
             break;
          case 6:
             printf("Enter the number: \n");
             scanf("%d", &number);
             rSquare2(number, &result);
             printf("rSquare2(): %d\n", result);
             break;
          default: printf("Program terminating .....\n");
             break;
       }
    } while (choice < 7);
    return 0;
}
int rNumDigits1(int num)
{
    if (num < 10)
       return 1;
    else
       return rNumDigits1(num/10) + 1;
}
void rNumDigits2(int num, int *result)
{
    /* Write your program code here */
}
int rDigitPos1(int num, int digit)
{
    /* Write your program code here */
}
void rDigitPos2(int num, int digit, int *pos)
{
    if (num % 10 == digit)
       *pos = 1;
    else if (num < 10)
       *pos = 0;
    else {
       rDigitPos2(num/10, digit, pos);
       if (*pos > 0)
          *pos += 1;
       else
          *pos = 0;
    }
}
int rSquare1(int num)
{
    /* Write your program code here */
}
void rSquare2(int num, int *result)
{
    /* Write your program code here */
}
```

Figure 1

1. (**rNumDigits**) Write a **recursive** function that counts the number of digits for a non-negative integer. For example, 1234 has 4 digits. Write two versions of the function. The function `rNumDigits1()` returns the result. The function `rNumDigits2()` returns the result through the parameter *`result`*. The function prototypes are given as follows:

```
int rNumDigits1(int num);
void rNumDigits2(int num, int *result);
```

For separate program testing: The following sample program template is given for testing the functions:

```
#include <stdio.h>
int rNumDigits1(int num);
void rNumDigits2(int num, int *result);
int main()
{
   int number, result=0;

   printf("Enter the number: \n");
   scanf("%d", &number);
   printf("rNumDigits1(): %d\n", rNumDigits1(number));
   rNumDigits2(number, &result);
   printf("rNumDigits2(): %d\n", result);
   return 0;
}
int rNumDigits1(int num)
{
   /* Write your program code here */
}
void rNumDigits2(int num, int *result)
{
   /* Write your program code here */
}
```

Some sample input and output sessions are given below:

(1) Test Case 1:
```
Enter the number:
5
rNumDigits1(): 1
rNumDigits2(): 1
```

(2) Test Case 2:
```
Enter the number:
13579
rNumDigits1(): 5
rNumDigits2(): 5
```

(3) Test Case 3:
```
Enter the number:
12
rNumDigits1(): 2
rNumDigits2(): 2
```

(4) Test Case 4:
```
Enter the number:
2468
rNumDigits1(): 4
rNumDigits2(): 4
```

2. (**rDigitPos**) Write a **recursive** function that returns the position of the first appearance of a specified digit in a positive number. The position of the digit is counted from the right and starts from 1. If the required digit is not in the number, the function should return 0. Write two versions of the function. The function `rDigitPos1()` returns the result. The function `rDigitPos2()` returns the result through the pointer parameter *pos*. The function prototypes are given as follows:

```
int rDigitPos1(int num, int digit);
void rDigitPos2(int num, int digit, int *pos);
```

For separate program testing: The following sample program template is given for testing the functions:

```
#include <stdio.h>
int rDigitPos1(int num, int digit);
void rDigitPos2(int num, int digit, int *pos);
int main()
{
   int number, digit, result=0;

   printf("Enter the number: \n");
   scanf("%d", &number);
   printf("Enter the digit: \n");
   scanf("%d", &digit);
   printf("rDigitPos1(): %d\n", rDigitPos1(number, digit));
   rDigitPos2(number, digit, &result);
   printf("rDigitPos2(): %d\n", result);
   return 0;
}
int rDigitPos1(int num, int digit)
{
   /* Write your program code here */
}
void rDigitPos2(int num, int digit, int *pos)
{
   /* Write your program code here */
}
```

Some sample input and output sessions are given below:

(1) Test Case 1:
```
Enter the number:
1234567
Enter the digit:
6
rDigitPos1(): 2
rDigitPos2(): 2
```

(2) Test Case 2:
```
Enter the number:
1234567
Enter the digit:
8
rDigitPos1(): 0
rDigitPos2(): 0
```

(3) Test Case 3:
```
Enter the number:
1357
Enter the digit:
3
rDigitPos1(): 3
rDigitPos2(): 3
```

(4) Test Case 4:
```
Enter the number:
6
Enter the digit:
6
rDigitPos1(): 1
rDigitPos2(): 1
```

3. (**rSquare**) Write a **recursive** function that returns the square of a positive integer number `num`, by computing the sum of odd integers starting with 1. The result is returned to the calling function. For example, if `num` = 4, then $4^2 = 1 + 3 + 5 + 7 = 16$ is returned; if `num` = 5, then $5^2 = 1 + 3 + 5 + 7 + 9 =$

25 is returned. Write two versions of the function. The function `rSquare1()` returns the result. The function `rSquare2()` returns the result through the parameter *`result`*. The function prototypes are:

```
int rSquare1(int num);
void rSquare2(int num, int *result);
```

For separate program testing: The following sample program template is given for testing the functions:

```
#include <stdio.h>
int rSquare1(int num);
void rSquare2(int num, int *result);
int main()
{
   int number, result=0;

   printf("Enter the number: \n");
   scanf("%d", &number);
   printf("rSquare1(): %d\n", rSquare1(number));
   rSquare2(number, &result);
   printf("rSquare2(): %d\n", result);
   return 0;
}
int rSquare1(int num)
{
   /* Write your program code here */
}
void rSquare2(int num, int *result)
{
   /* Write your program code here */
}
```

Some sample input and output sessions are given below:

(1) Test Case 1:
```
Enter a number:
4
rSquare1(): 16
rSquare2(): 16
```

(2) Test Case 2:
```
Enter a number:
1
rSquare1(): 1
rSquare2(): 1
```

(3) Test Case 3:
```
Enter a number:
12
rSquare1(): 144
rSquare2(): 144
```

(4) Test Case 4:
```
Enter a number:
5
rSquare1(): 25
rSquare2(): 25
```