WEEK 7 - RECURSIVE FUNCTIONS

You are required to do the following:

1. Lab Questions – please do the lab questions during the lab session.
2. Assignment Questions – please do the assignment questions and submit your code to the online automated programming assignment submission system (APAS) for grading.

Lab Tutor: For this lab-tutorial session, please discuss about the solution for each question in the lab. You may allocate about 30 minutes for each question. No need to discuss about the assignment questions.

Lab Questions

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:

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

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
```

For separate program testing: The following sample program template is given below:

```
#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 */
}
```

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:

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

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
```

For separate program testing: The following sample program template is given below:

```
#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 */
}
```

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);
```

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
```

For separate program testing: The following sample program template is given below:

```
#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 */
}
```

4. Predict the output from the program below:

```
#include  <stdio.h>
#define BLANK ‘ ’

void saveChar(void);

int main()
{
   printf(“Enter your word and end it with a space =>”);
   saveChar();
   putchar(‘\n’);

   return 0;
}
void saveChar()
{
   char ch;

   ch = getchar();
   if (ch != BLANK)
      saveChar();
   else
      putchar(‘\n’);
   putchar(ch);
}
```

Assignment Questions for Submission

You are required to submit your code for the following assignment questions to the Automated Programming Submission and Assessment System (APAS) for marking and grading. For example, if your lab session is held on the week of 14 August 2017 (any day of the week), then the deadline for your submission will be on Friday the following week 25 August 2017 (Friday), at 11.59pm.

1. (**rCountZeros**) Write a **recursive** C function that counts the number of zeros in a specified positive number num. For example, if *num* is 105006, then the function will return 3; and if *num* is 1357, the function will return 0. Write the function in two versions. The function `rCountZeros1()` returns the result, while the function `rCountZeros2()` passes the result through the parameter *result*. The function prototypes are given as follows:

```
int rCountZeros1(int num);
void rCountZeros2(int num, int *result);
```

Write a C program to test the functions.

Some sample input and output sessions are given below:

(1) Test Case 1:
```
Enter the number:
10500
rCountZeros1(): 3
rCountZeros2(): 3
```

(2) Test Case 2:
```
Enter the number:
23453
rCountZeros1(): 0
rCountZeros2(): 0
```

(3) Test Case 3:
```
Enter the number:
0
rCountZeros1(): 1
rCountZeros2(): 1
```

A sample program to test the functions is given below:

```
#include <stdio.h>
int rCountZeros1(int num);
void rCountZeros2(int num, int *result);
int main()
{
   int number, result;

   printf("Enter the number: \n");
   scanf("%d", &number);
   printf("rCountZeros1(): %d\n", rCountZeros1(number));
   rCountZeros2(number,&result);
   printf("rCountZeros2(): %d\n", result);
   return 0;
}
int rCountZeros1(int num)
{
   /* Write your program code here */
}
void rCountZeros2(int num, int *result)
{
   /* Write your program code here */
}
```

2. **(rStrLen)** The following **recursive** function `rStrLen()` accepts a character string *s* as parameter, and returns the length of the string. For example, if *s* is `"abcde"`, then the function `rStrLen()` will return 5. The function prototype is:

```
int rStrLen(char *s);
```

Write a C program to test the function.

Some sample input and output sessions are given below:

(1) Test Case 1:
```
Enter the string:
abc de
rStrLen(): 6
```

(2) Test Case 2:
```
Enter the string:
a
rStrLen(): 1
```

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

```
#include <stdio.h>
int rStrLen(char *s);
int main()
{
   char str[80];

   printf("Enter the string: \n");
   gets(str);
   printf("rStrLen(): %d\n", rStrLen(str));
   return 0;
}
int rStrLen(char *s)
{
   /* Write your program code here */
}
```

3. (**rReverseAr**) Write a **recursive** function whose arguments are an array of integers *ar* and an integer *size* specifying the size of the array and whose task is to reverse the contents of the array. The result is returned to the caller through the array parameter. The code should not use another, intermediate, array. The function prototype is given as follows:

```
void rReverseAr(int ar[], int size);
```

Write a C program to test the function.

Some sample input and output sessions are given below:

(1) Test Case 1:
```
Enter size:
5
Enter 5 numbers:
1 2 3 4 5
rReverseAr(): 5 4 3 2 1
```

(2) Test Case 2:
```
Enter size:
1
Enter 1 numbers:
5
rReverseAr(): 5
```

A sample program to test the function is given below:

```
#include <stdio.h>
void rReverseAr(int ar[], int size);
int main()
```

```
{
   int array[80];
   int size, i;

   printf("Enter size: \n", &size);
   scanf("%d", &size);
   printf("Enter %d numbers: \n", size);
   for (i = 0; i < size; i++)
      scanf("%d", &array[i]);
   printf("rReverseAr(): ");
   rReverseAr(array, size);
   for (i = 0; i < size; i++)
      printf("%d ", array[i]);
   printf("\n");
   return 0;
}
void rReverseAr(int ar[], int size)
{
   /* Write your program code here */
}
```

4. (**rCountArray**) Write a **recursive** C function `rCountArray()` that returns the number of times the integer *a* appears in the array which has *n* integers in it. Assume that *n* is greater than or equal to 1. The function prototype is:

```
int rCountArray(int array[], int n, int a);
```

Write a C program to test the function.

Some sample input and output sessions are given below:

(1) Test Case 1:

```
Enter array size:
10
Enter 10 numbers:
1 2 3 4 5 5 6 7 8 9
Enter the target number:
5
rCountArray(): 2
```

(2) Test Case 2:

```
Enter array size:
5
Enter 5 numbers:
1 2 3 4 5
Enter the target number:
8
rCountArray(): 0
```

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

```
#include <stdio.h>
#define SIZE 20
int rCountArray(int array[], int n, int a);
int main()
{
   int array[SIZE];
   int index, count, target, size;

   printf("Enter array size: \n");
   scanf("%d", &size);
   printf("Enter %d numbers: \n", size);
   for (index = 0; index < size; index++)
      scanf("%d", &array[index]);
```

```
    printf("Enter the target number: \n");
    scanf("%d", &target);
    count = rCountArray(array, size, target);
    printf("rCountArray(): %d\n", count);
    return 0;
}
int rCountArray(int array[], int n, int a)
{
    /* Write your program code here */
}
```