Section D – Recursive Funcitions [Ans 1 Specified Qn from this Section]

1. (**rNumDigits1**) Write a **recursive** function that counts the number of digits for a non-negative integer. For example, 1234 has 4 digits. The function `rNumDigits1()` returns the result. The function prototype of the function is given below:

```
int rNumDigits1(int num);
```

Some sample input and output sessions are given below:

(1) Test Case 1:

```
Enter the number:
1
rNumDigits1(): 1
```

(2) Test Case 2:

```
Enter the number:
13579
rNumDigits1(): 5
```

A sample program to test the functions is given below:

```
#include <stdio.h>
int rNumDigits1(int num);
int main()
{
   int number;

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

2. (**rDigitPos2**) 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. The function `rDigitPos2()` returns the result through the pointer parameter *pos*. The function prototype is given below:

```
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
rDigitPos2(): 2
```

(2) Test Case 2:

```
Enter the number:
1234567
Enter the digit:
8
rDigitPos2(): 0
```

A sample program to test the functions is given below:

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

   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);
   return 0;
}
void rDigitPos2(int num, int digit, int *pos)
{
   /* Write your program code here */
}
```

3. (**rSquare1**) 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. The function `rSquare1()` returns the result. The function prototype is given below:

```
int rSquare1(int num);
```

Some sample input and output sessions are given below:

(1) Test Case 1:

```
Enter a number:
4
rSquare1(): 16
```

(2) Test Case 2:

```
Enter a number:
1
rSquare1(): 1
```

A sample program to test the functions is given below:

```
#include <stdio.h>
int rSquare1(int num);
int main()
{
   int x;

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

4. **(rStrLen)** The **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:
abcde
rStrLen(): 5
```

(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 */
}
```

5. (**rCountZeros2**) 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. The function `rCountZeros2()` passes the result through the pointer parameter *result*. The function prototype is given as follows:

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

Write a C program to test the function.

Some sample input and output sessions are given below:

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

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

(3) Test Case 3:

```
Enter the number:
0
rCountZeros2(): 1
```

A sample program to test the functions is given below:

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

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

6. (**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:
3
rReverseAr(): 3
```

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 */
}
```

7. (**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:
5
rCountArray(): 2
```

(2) Test Case 2:

```
Enter array size:
5
Enter 5 numbers:
1 2 3 4 5
Enter the target:
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: \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 */
```

```
}
```