Tutorial 6 – Recursive Functions

1. (**rSumup**) A function **rSumup()** is defined as:

   rSumup(1) = 1
   rSumup(n) = n + rSumup(n-1) if n > 1

   Implement rSumup() in two versions. The function rSumup1() computes and returns the result. The function rSumup2() computes and returns the result through the parameter result. The function prototypes are given as follows:

   ```
   int rSumup1(int n);
   void rSumup2(int n, int *result);
   ```

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

   ```
   #include <stdio.h>
   int rSumup1(int n);
   void rSumup2(int n, int *result);
   int main()
   {
     int n, result;

     printf("Enter a number: \n");
     scanf("%d", &n);
     printf("rSumup1(): %d\n", rSumup1(n));
     rSumup2(n, &result);
     printf("rSumup2(): %d\n",result);
     return 0;
   }
   int rSumup1(int n)
   {
     /* Write your code here */
   }
   void rSumup2(int n, int *result)
   {
     /* Write your code here */
   }
   ```

   Some sample input and output sessions are given below:

   (1) Test Case 1:
   Enter a number:
   5
   rSumup1(): 15
   rSumup2(): 15

   (2) Test Case 2:
   Enter a number:
   10
   rSumup1(): 55
   rSumup2(): 55

2. (**rDigitValue**) Write a **recursive** function that returns the value of the k^{th} digit (k>0) from the right of a non-negative integer num. For example, if num is12348567 and k is 3, the function will return 5 and if num is 1234 and k is 8, the function will return 0. Write the recursive function in two versions. The function rDigitValue1() computes and returns the result. The function rDigitValue2() computes and returns the result through the parameter result using call by reference. The function prototypes are given below:

```
int rDigitValue1(int num, int k);
void rDigitValue2(int num, int k, int *result);
```

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

```
#include <stdio.h>
int rDigitValue1(int num, int k);
void rDigitValue2(int num, int k, int *result);
int main()
{
  int k;
  int number, digit;

  printf("Enter a number: \n");
  scanf("%d", &number);
  printf("Enter k position: \n");
  scanf("%d", &k);
  printf("rDigitValue1(): %d\n", rDigitValue1(number, k));
  rDigitValue2(number, k, &digit);
  printf("rDigitValue2(): %d\n", digit);
  return 0;
}
int rDigitValue1(int num, int k)
{
  /* Write your code here */
}
void rDigitValue2(int num, int k, int *result)
{
  /* Write your code here */
}
```

Some sample input and output sessions are given below:

(1) Test Case 1:

```
Enter a number:
2348567
Enter k position:
3
rDigitValue1(): 5
rDigitValue2(): 5
```

(2) Test Case 2:

```
Enter a number:
```

123
Enter k position:
4
rDigitValue1(): 0
rDigitValue2(): 0

(3) Test Case 3:
Enter a number:
12456
Enter k position:
1
rDigitValue1(): 6
rDigitValue2(): 6

(4) Test Case 4:
Enter a number:
82345
Enter k position:
5
rDigitValue1(): 8
rDigitValue2(): 8

3. 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);
}
```

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);
```

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

```
#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 */
}
```

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
```

(3) Test Case 3:

```
Enter array size:
1
Enter 1 numbers:
5
Enter the target number:
5
rCountArray(): 1
```

(4) Test Case 4:

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