Recursive Functions – Q1

Write a 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);
```

```
#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)
{
   if (num < 10)
      return 1;
   else
      return rNumDigits1(num/10) + 1;
}
```

Enter the number:
1234
rNumDigits1(): 4

Enter the number:
13579
rNumDigits2(): 5

Enter the number:
1234
rNumDigits1(): 4

Enter the number:
13579
rNumDigits2(): 5

Note:

When dealing with numbers, the integer division operator and modulus operator can be used to extract the digit value from the number.

Recursive Functions – Q1

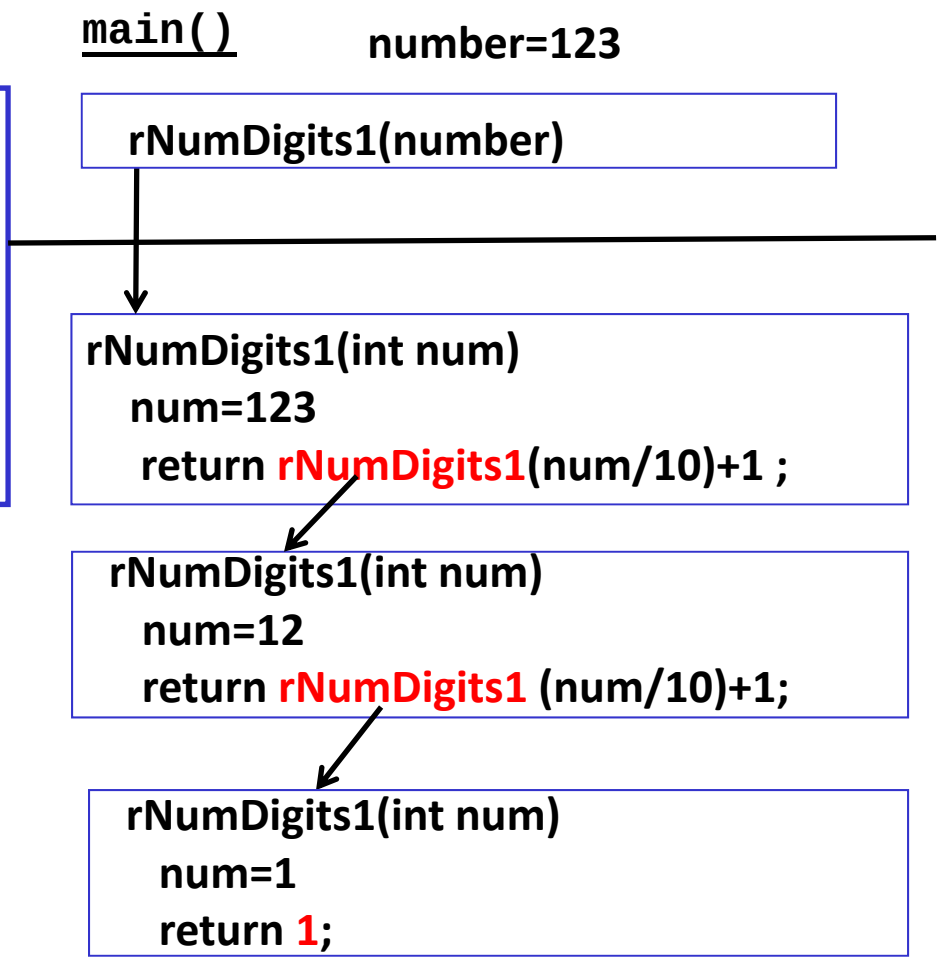
By Returning Value

```
int rNumDigits1(int num)
{
   if (num < 10)
      return 1;
   else
      return rNumDigits1(num/10)+1;
}
```

Enter the number:

123

rNumDigits1(): 3



Recursive Functions – Q1

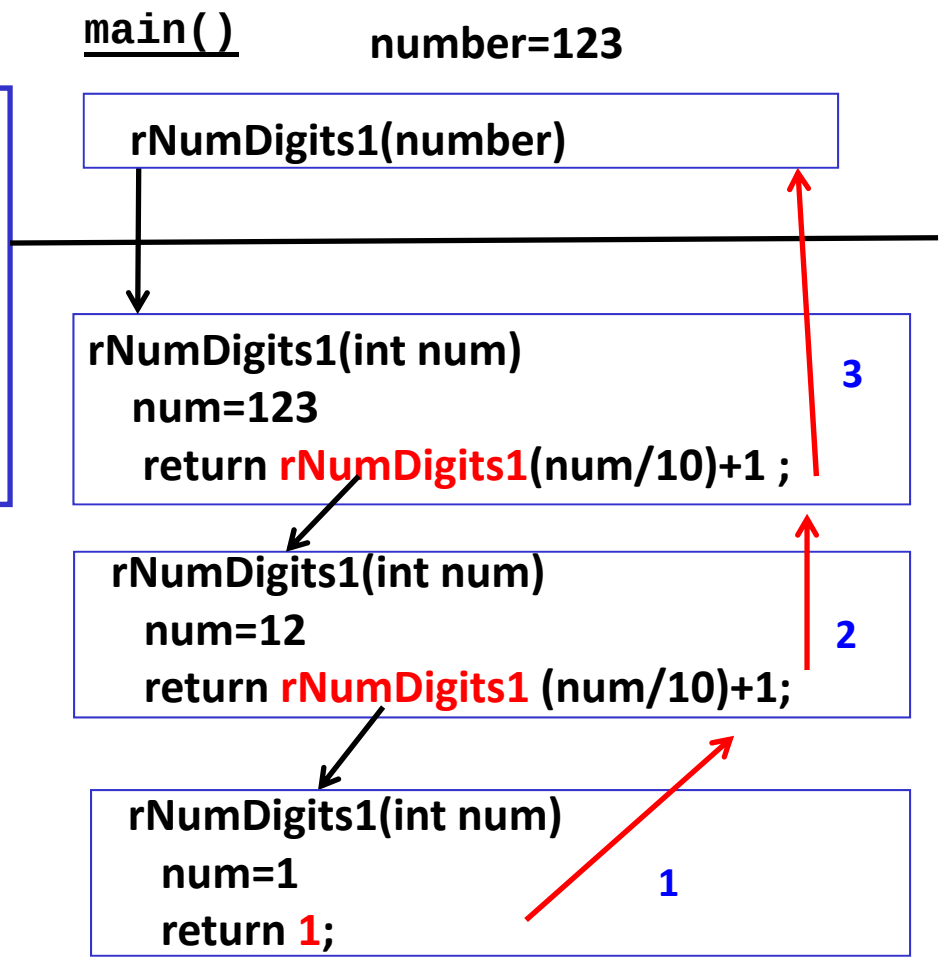
By Returning Value

```
int rNumDigits1(int num)
{
   if (num < 10)
      return 1;
   else
      return rNumDigits1(num/10)+1;
}
```

Enter the number:

123

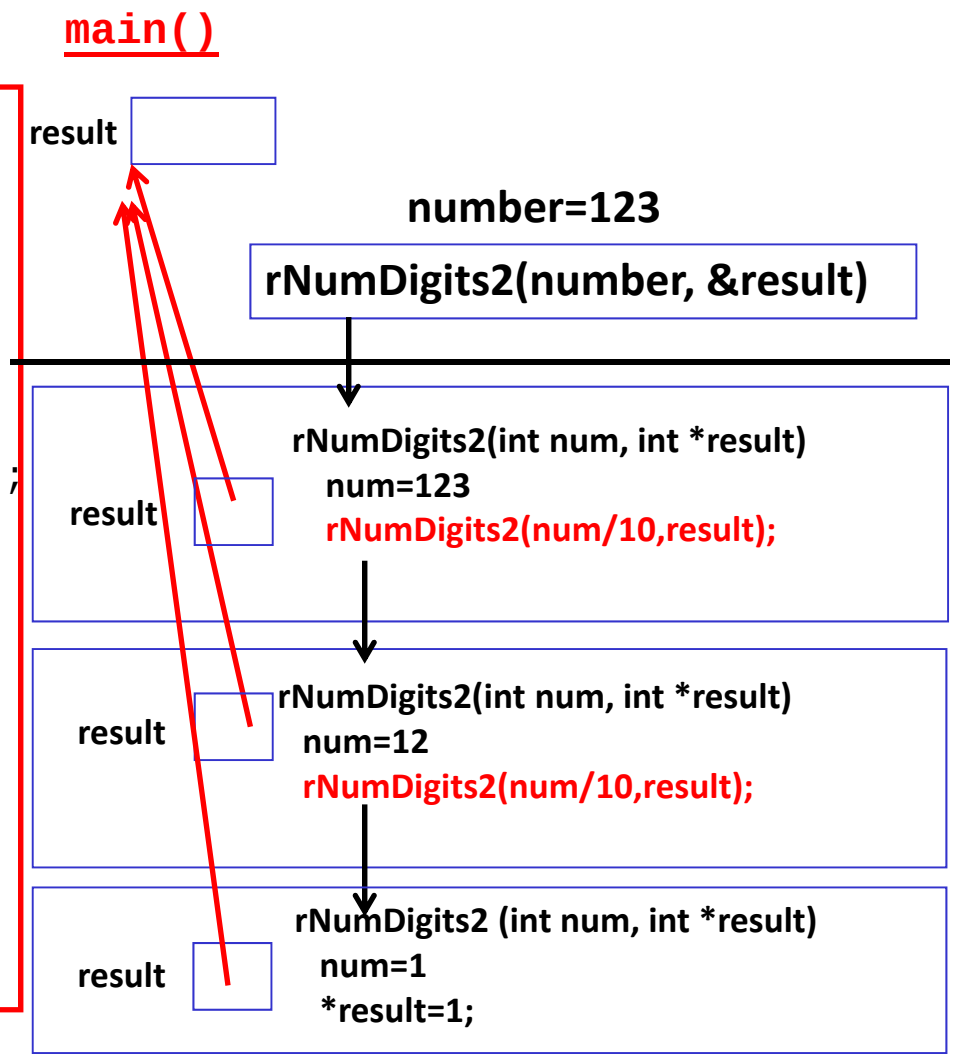
rNumDigits1(): 3



Recursive Functions – Q1

Call by reference

```
#include <stdio.h>
void rNumDigits2(int num, int *result);
int main()
{
   int number, result;
   printf("Enter the number: \n");
   scanf("%d", &number);
   rNumDigits2(number, &result);
   printf("rNumDigits2(): %d\n", result);
   return 0;
}
void rNumDigits2(int num, int *result)
{
   if (num < 10)
      *result = 1;
   else {
      rNumDigits2(num/10, result);
      *result = *result + 1;
   }
}
```



Enter the number:

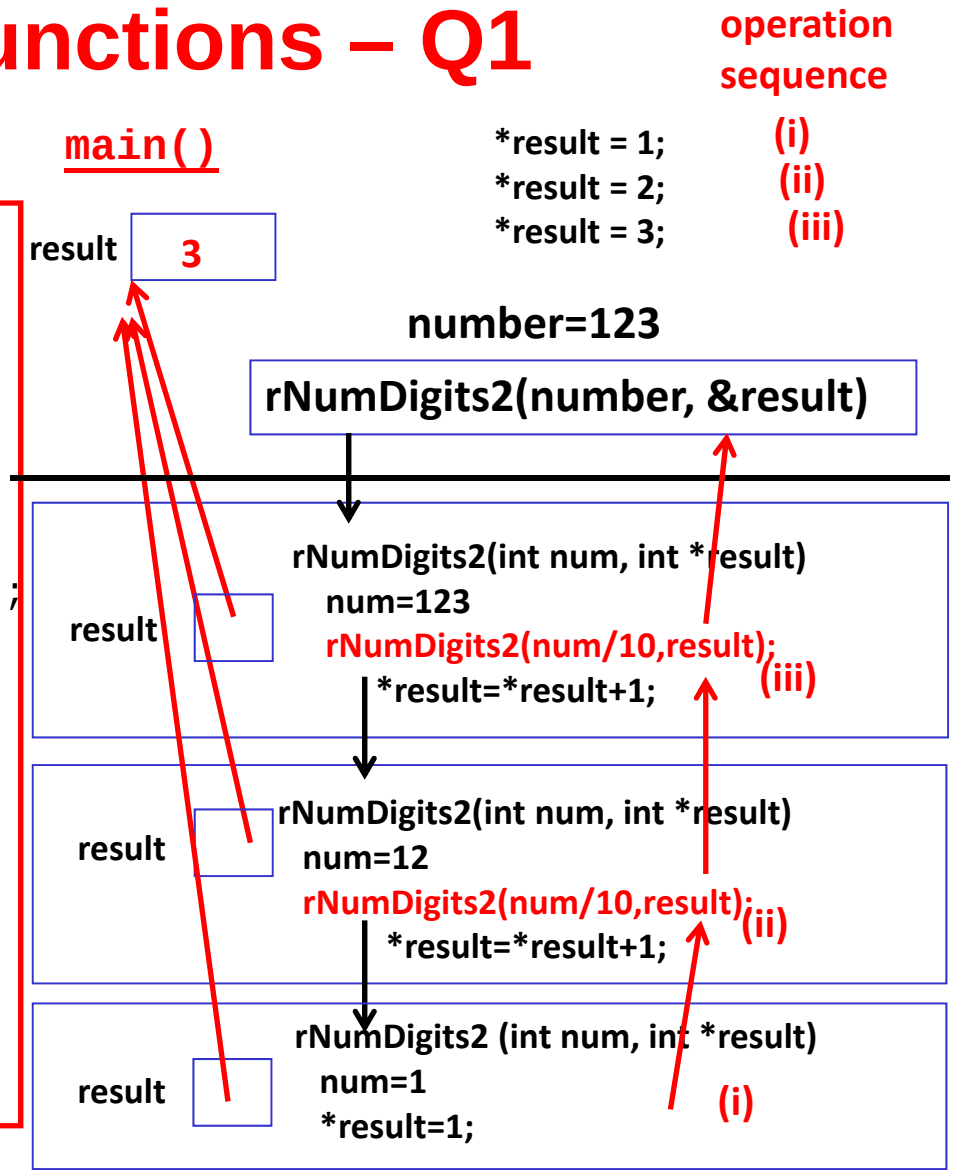
123

rNumDigits2(): 3

Recursive Functions – Q1

Call by reference

```
#include <stdio.h>
void rNumDigits2(int num, int *result);
int main()
{
   int number, result;
   printf("Enter the number: \n");
   scanf("%d", &number);
   rNumDigits2(number, &result);
   printf("rNumDigits2(): %d\n", result);
   return 0;
}
void rNumDigits2(int num, int *result)
{
   if (num < 10)
      *result = 1;
   else {
      rNumDigits2(num/10, result);
      *result = *result + 1;
   }
}
```



Enter the number:

123

rNumDigits2(): 3

Recursive Functions – Q2

Write a 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 parameter *result*. The function prototypes are:

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

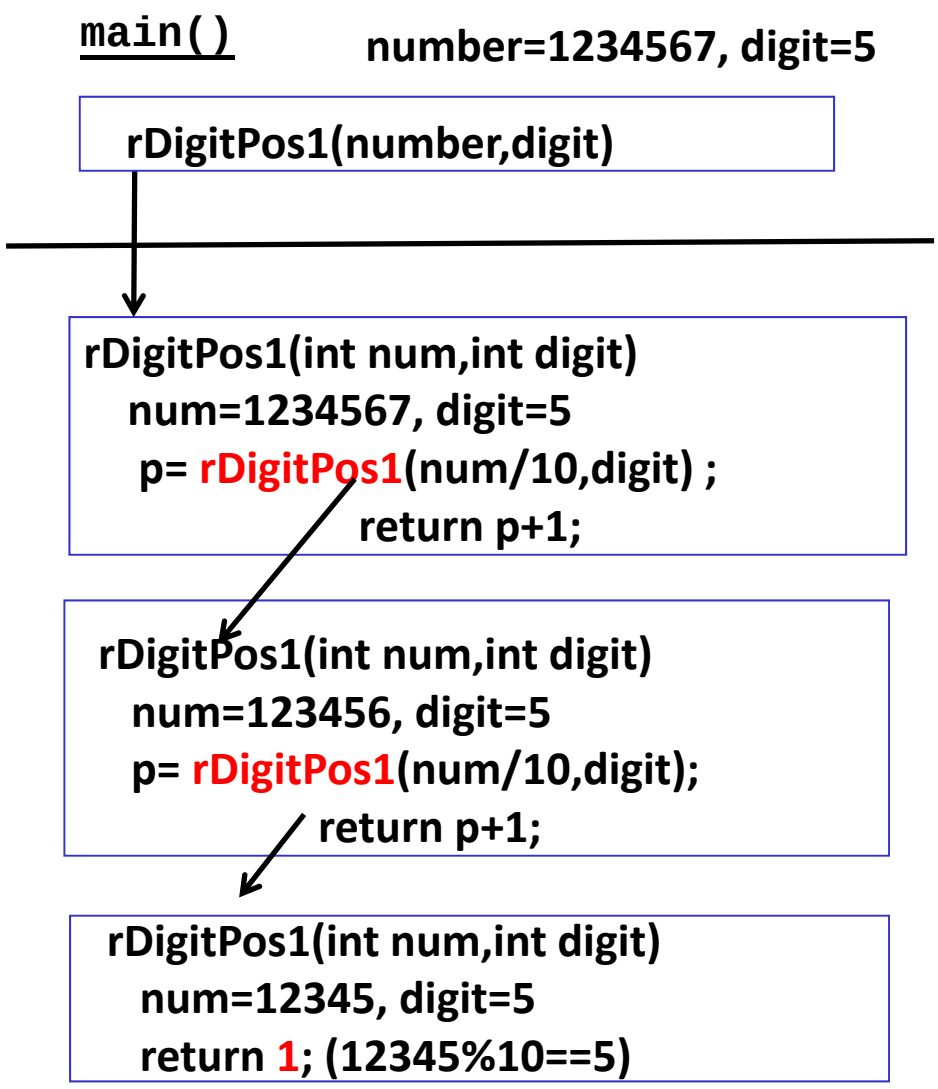
Enter the number:
1234567
Enter the digit:
6
rDigitPos1(): 2
Enter the number:
1234567
Enter the digit:
8
rDigitPos2(): 0

Recursive Functions – Q2

By Returning Value

```
#include <stdio.h>
int rDigitPos1(int num, int digit);
int main()
{
   int number, digit;
   printf("Enter the number: \n");
   scanf("%d", &number);
   printf("Enter the digit: \n");
   scanf("%d", &digit);
   printf("rDigitPos1(): %d\n",
       rDigitPos1(number, digit));
   return 0;
}
```

```
int rDigitPos1(int num, int digit)
{
   int p;
   if (num % 10 == digit)
      return 1;
   else if (num < 10)
      return 0;
   else {
      p = rDigitPos1(num/10, digit);
      if (p > 0)
         return p + 1;
      else
         return 0;
   }
}
```



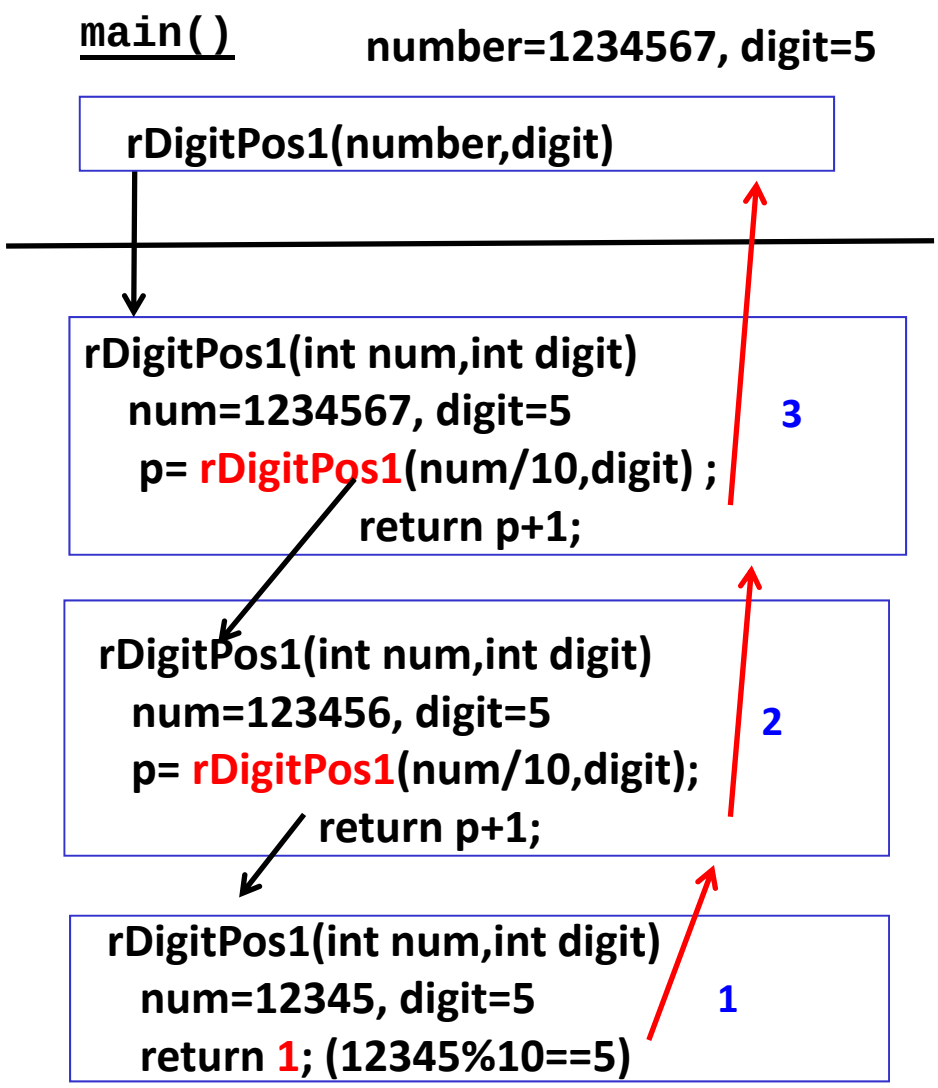
Enter the number:
1234567
Enter the digit:
5
rDigitPos1(): 3

Recursive Functions – Q2

By Returning Value

```
#include <stdio.h>
int rDigitPos1(int num, int digit);
int main()
{
   int number, digit;
   printf("Enter the number: \n");
   scanf("%d", &number);
   printf("Enter the digit: \n");
   scanf("%d", &digit);
   printf("rDigitPos1(): %d\n",
       rDigitPos1(number, digit));
   return 0;
}
```

```
int rDigitPos1(int num, int digit)
{
   int p;
   if (num % 10 == digit)
      return 1;
   else if (num < 10)
      return 0;
   else {
      p = rDigitPos1(num/10, digit);
      if (p > 0)
         return p + 1;
      else
         return 0;
   }
}
```



Enter the number:

1234567

Enter the digit:

5

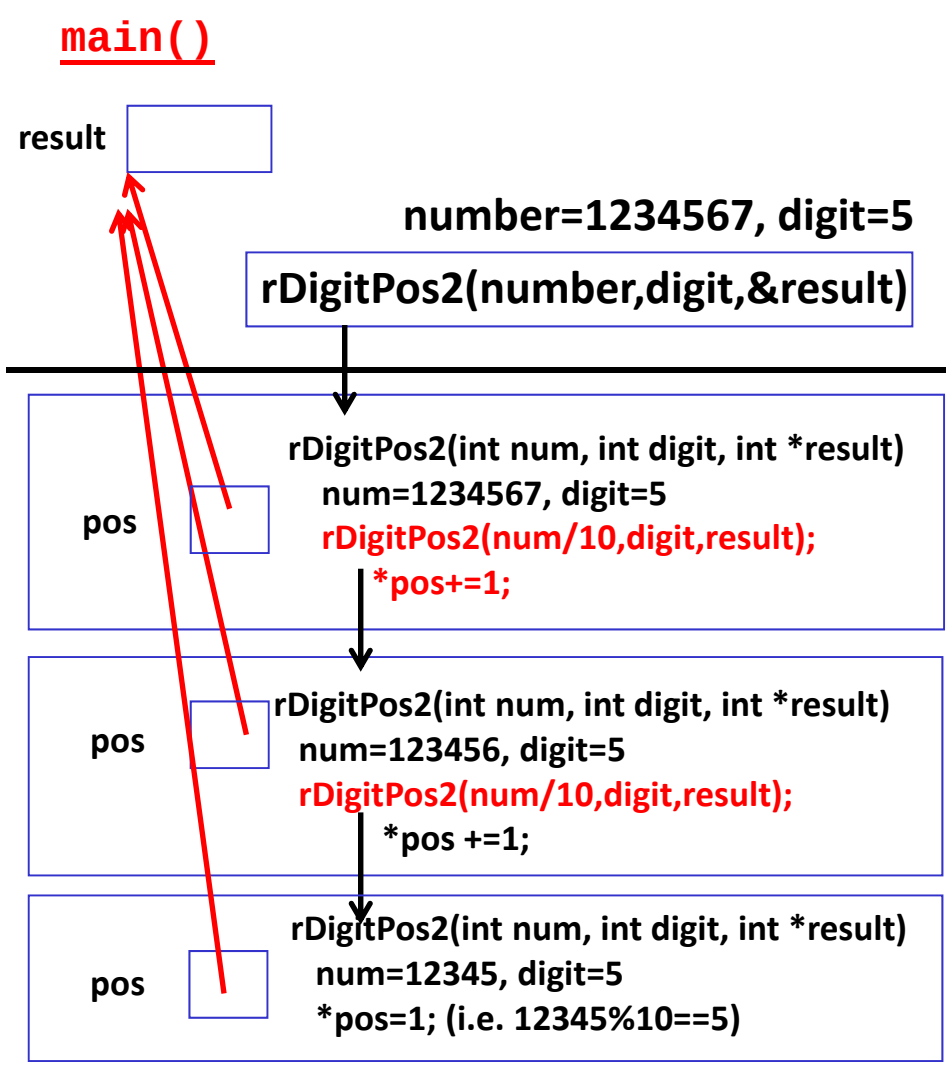
rDigitPos1(): 3

Recursive Functions – Q2

Call by reference

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



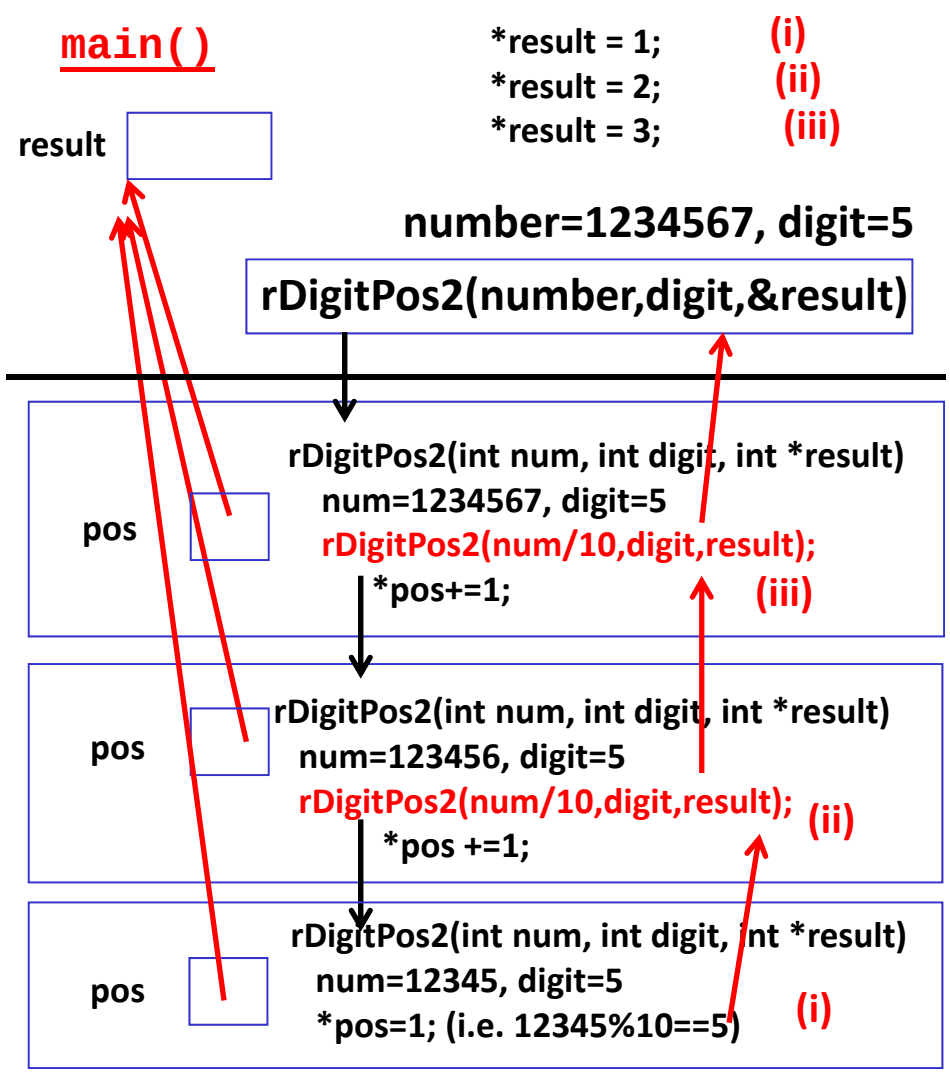
Enter the number:
1234567
Enter the digit:
5
rDigitPos1(): 3

Recursive Functions – Q2

Call by reference

```
#include <stdio.h>
void rDigitPos2(int num, int digit, int
*pos);
int main()
{
   int number;
   int digit, result=0;
   printf("Enter the number: ");
   scanf("%d", &number);
   printf("Enter the digit: ");
   scanf("%d", &digit);
   rDigitPos2(number, digit, &result);
   printf("rDigitPos2(): %d", result);
   return 0;
}
```

```
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;
   }
}
```



Enter the number:
1234567
Enter the digit:
5
rDigitPos1(): 3

Recursive Functions – Q3

Write a 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);
```

Enter a number:

4

rSquare1(): 16

Enter a number:

5

rSquare2(): 25

Recursive Functions – Q3

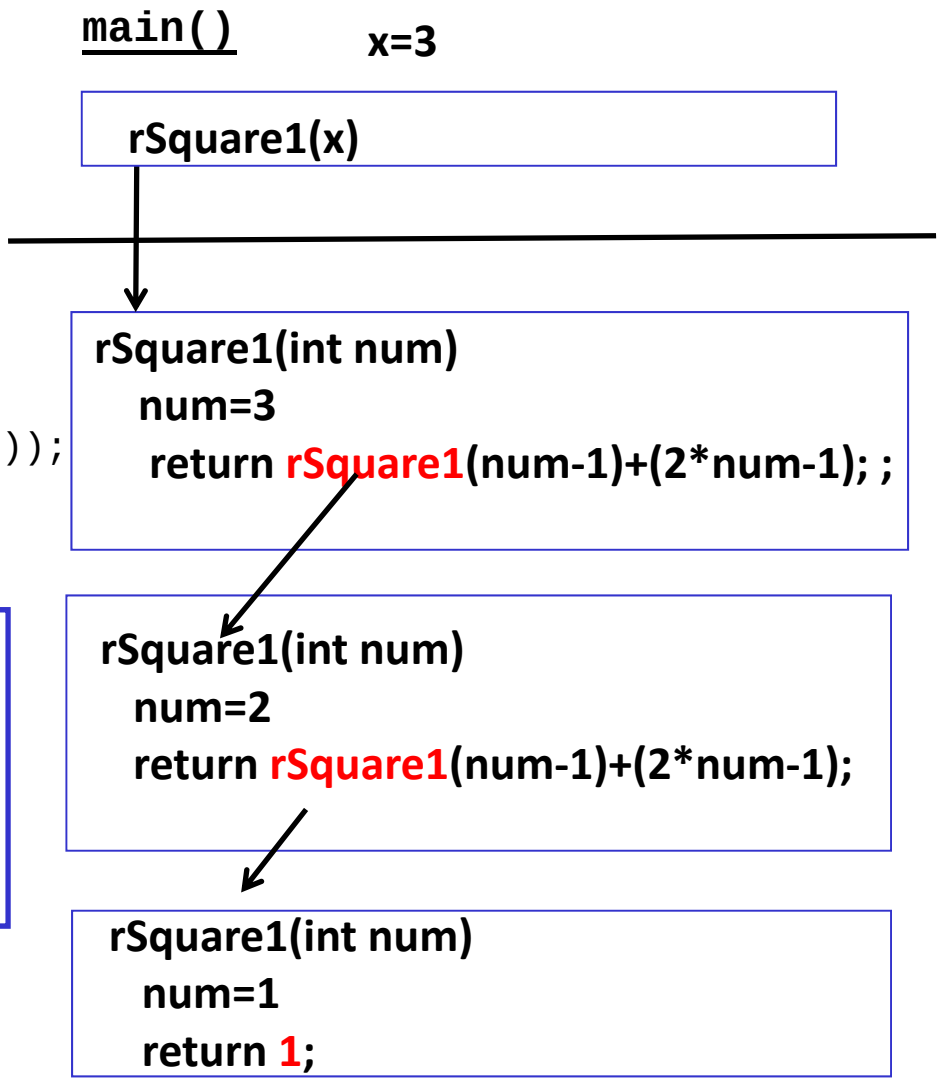
By Returning Value

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

int main()
{
   int x;

   printf("Enter a number: \n");
   scanf("%d", &x);
   printf("rSquare1(): %d\n", rSquare1(x));

   return 0;
}
int rSquare1(int num)
{
   if (num == 1)
      return 1;
   else
      return rSquare1(num-1)+(2*num -1);
}
```

Enter a number:

3

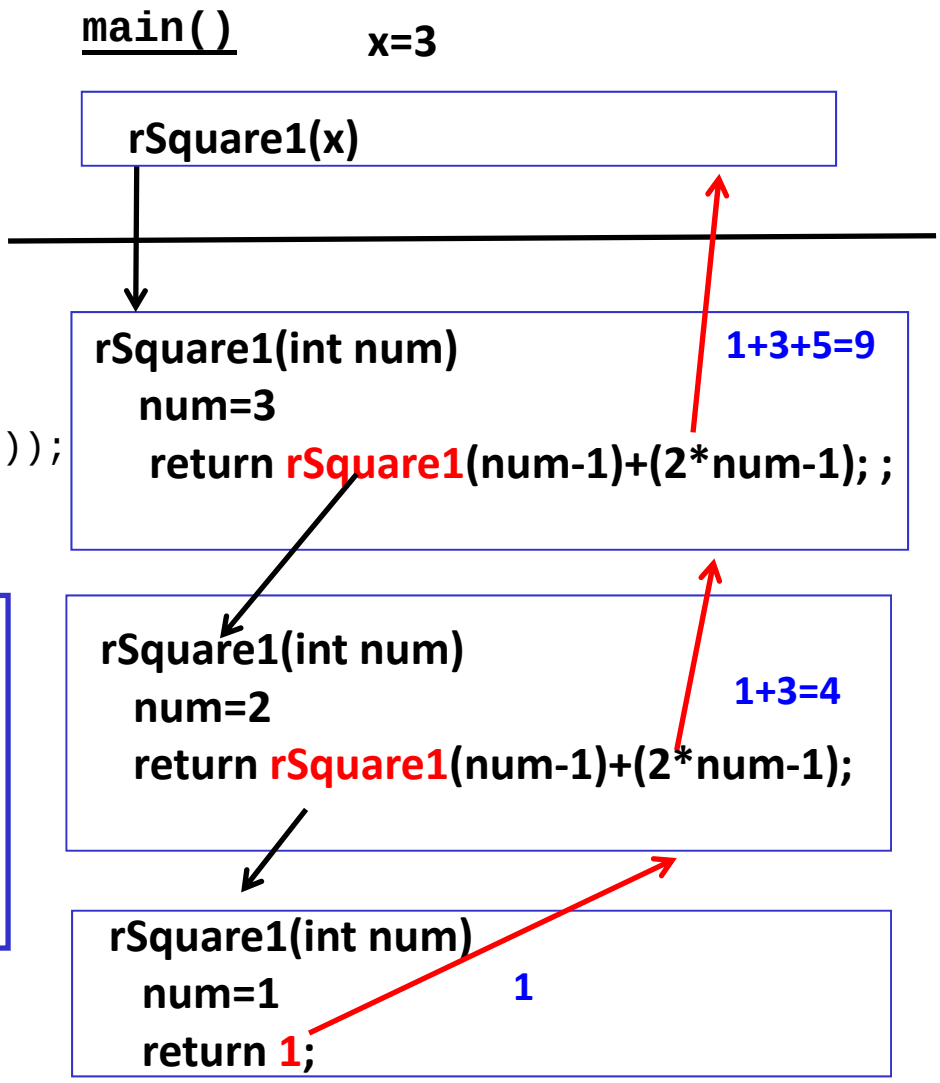
rSquare1(): 9



Recursive Functions – Q3

By Returning Value

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

int main()
{
   int x;

   printf("Enter a number: \n");
   scanf("%d", &x);
   printf("rSquare1(): %d\n", rSquare1(x));

   return 0;
}
int rSquare1(int num)
{
   if (num == 1)
      return 1;
   else
      return rSquare1(num-1)+(2*num -1);
}
```



Enter a number:

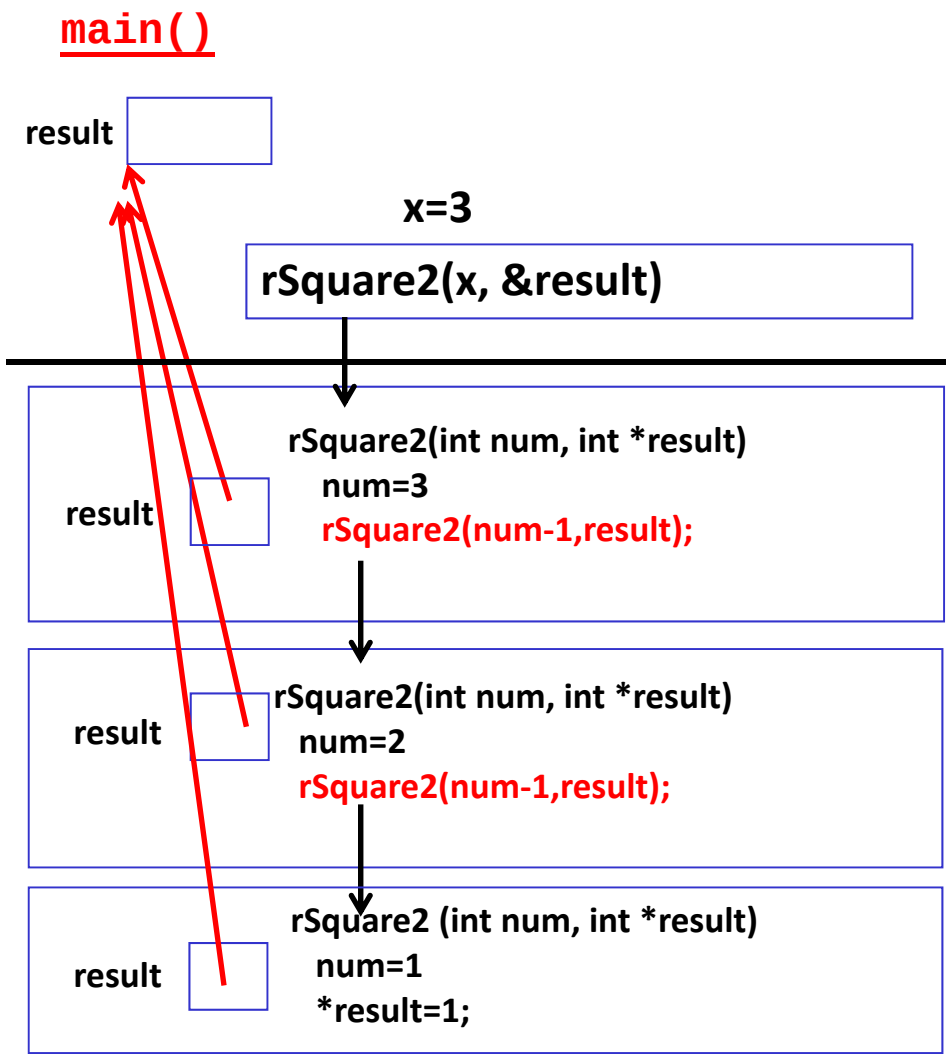
3

rSquare1(): 9

Recursive Functions – Q3

Call by reference

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

   printf("Enter a number: \n");
   scanf("%d", &x);
   rSquare2(x, &result);
   printf("rSquare2(): %d\n", result);
   return 0;
}
void rSquare2(int num, int *result)
{
   if (num == 1)
      *result = 1;
   else {
      rSquare2(num-1, result);
      *result += (2*num -1);
   }
}
```



Enter the number:

123

rNumDigits2(): 3

Recursive Functions – Q3

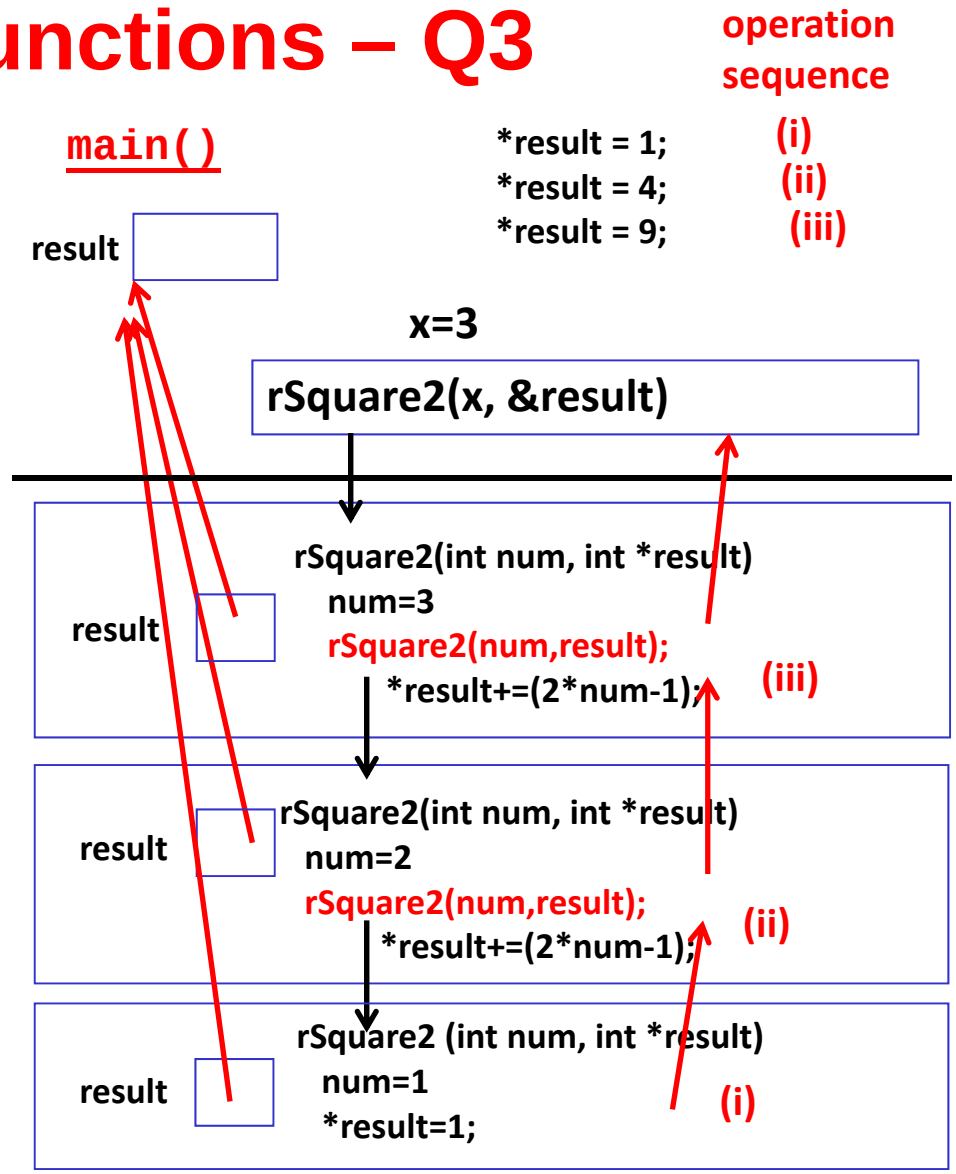
Call by reference

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

   printf("Enter a number: \n");
   scanf("%d", &x);
   rSquare2(x, &result);
   printf("rSquare2(): %d\n", result);
   return 0;
}
void rSquare2(int num, int *result)
{
   if (num == 1)
      *result = 1;
   else {
      rSquare2(num-1, result);
      *result += (2*num -1);
   }
}
```

Enter the number:

123

rNumDigits2(): 3



Recursive Functions – Q4

```
#include  <stdio.h>
#define BLANK ' '
void saveChar();
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);
}
```

Enter your word and end it with a space => **ward**

What is the output?

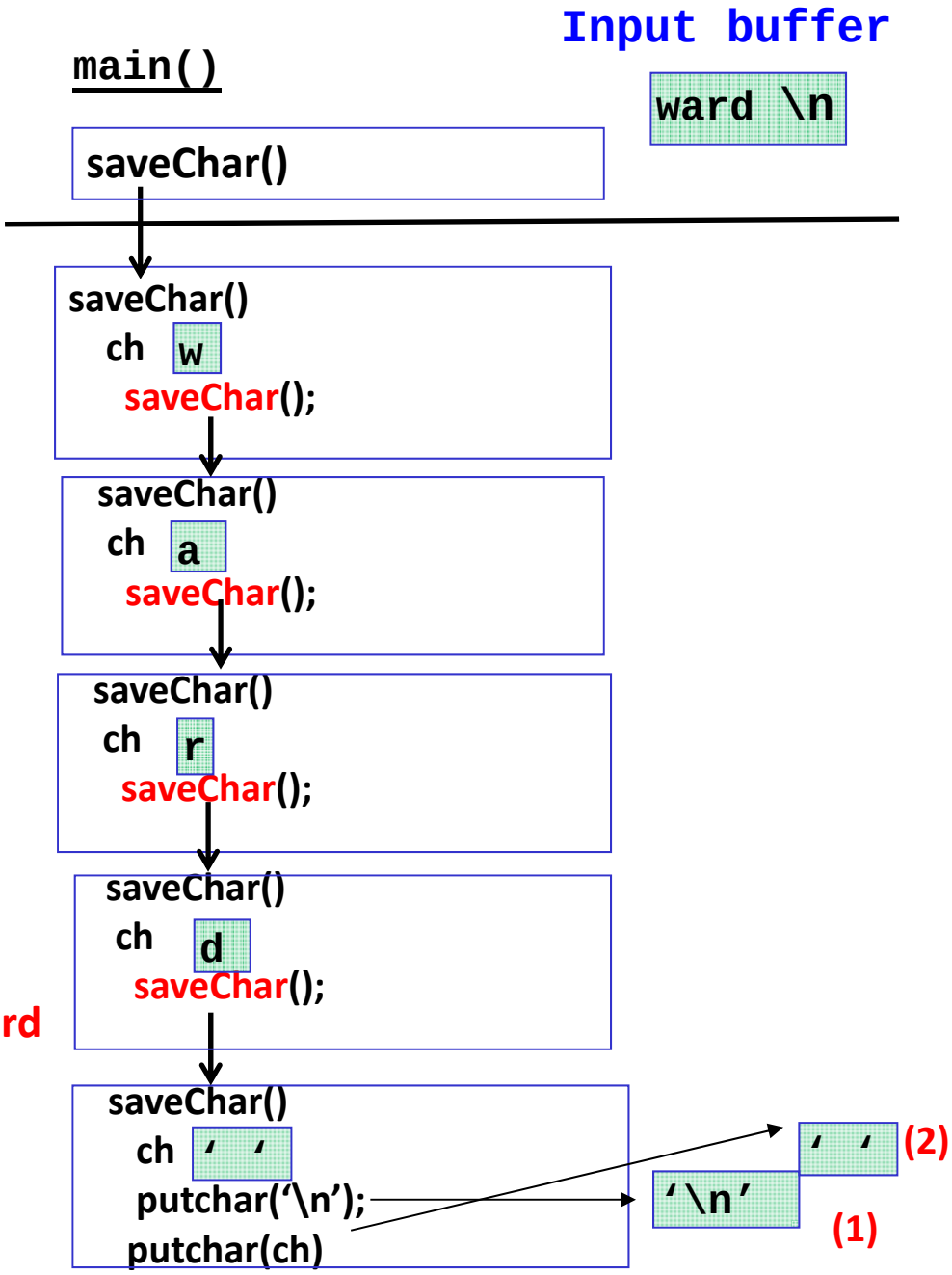
You may try to enter the code and run the code to see the output.

Recursive Functions – Q4

```
#include  <stdio.h>
#define BLANK ' '
void saveChar();
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);
}
```

Enter your word and end it with a space => **ward**

draw

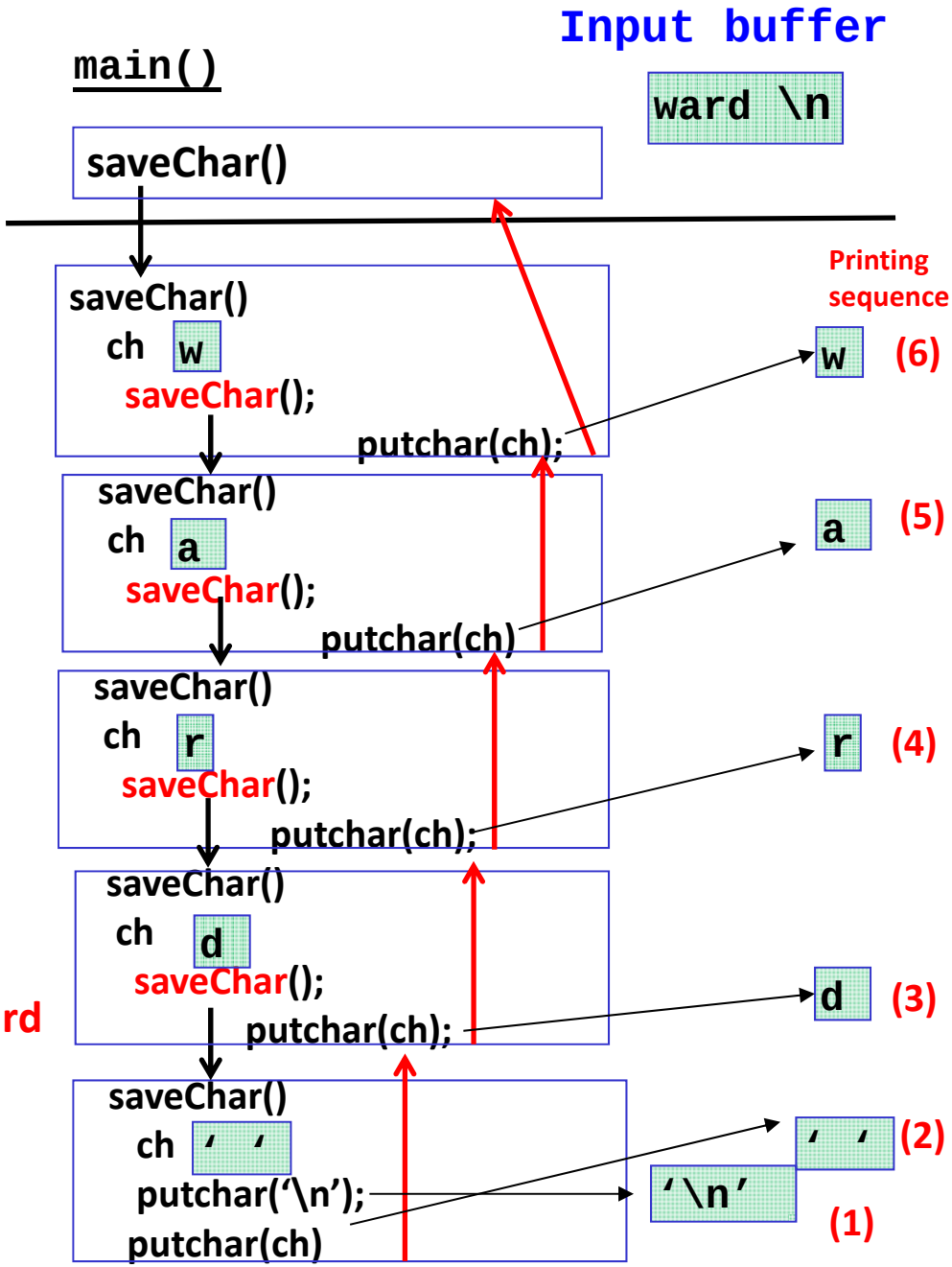


Recursive Functions – Q4

```
#include  <stdio.h>
#define BLANK ' '
void saveChar();
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);
}
```

Enter your word and end it with a space => **ward**

draw



Recursive Functions – Q4

```
#include  <stdio.h>
#define BLANK ' '
void saveChar();
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);
}
```

Enter your word and end it with a space => **ward**

draw

Please note that there is a blank character at the end of the input word before the “enter” key is pressed.

Basically, this program prints an input string, which ends with a space character, in the reversed order.