CS19001: Programming and Data Structures Laboratory

DRC, SD, SB, CSE, IIT Kharagpur

March 5, 2025

Iterative and Recursive Functions

Function

- A *function is a self-contained program segment that* carries out some specific, well-defined task.
- main is a function.
- Program execution begins from main
- There can be multiple functions. They can appear in any order.
- One function definition cannot be embedded within another.
- A function will carry out when it is “called”

Function

- A function will process information passed to it from the calling program
- Information is passed via special identifiers called arguments or parameters
- The value is returned via the return statement

Function Definition

- data-type name (argument1, argument 2, …. ……. argument n)

- return expression; // expression is optional

- A function can return only one value

Function

```
long int fact(n)

  int n;
   {
      int i;
      long int prod = 1;
      if (n > 1)
         for (i =2; i <= n; ++i)
              prod *= i;
      return(prod)
    }
```

Function

```
#include <stdio,h>
main()
  {
    int n;
    long int fact(); // function declaration
    printf("\n Enter the number ");
    scanf("%d", &n);
    printf("\n the factorial is %ld", fact(n));
  }
```

Recursion

- A process by which a function calls itself repeatedly
  - Either directly.
  - X calls X.
  - Or cyclically in a chain.
  - X calls Y, and Y calls X.
- Used for repetitive computations in which each action is stated in terms of a previous result
  - fact(n) = n * fact (n-1)

Recursive Function

- For a problem to be written in recursive form, two conditions are to be satisfied:

– It should be possible to express the problem in recursive form.

– The problem statement must include a stopping condition

fact(n) = 1, if n = 0

= n * fact(n-1), if n > 0

Sample Program

```
int factorial (int n)
 {
    int value;
    if ( n == 0) // TERMINATING CONDITION
       return 1;
    else          // this 'else' is optional
       value = n*factorial(n-1);          // RECURSIVE
                                  // FACTORIAL CALCULATUION
    return value;          // FINAL VALUE RETURN
}
```

Programming Assignments

Complete and submit during lab

Assignment 1

1(a). Write a C function fact(n) to compute the factorial of "n". Read n from keyboard and print it's factorial in the main program.

Write a program to compute the value of ${}^{n}C_{r}$ by using the above function fact(n).

Read n and r in the main program and print the inputs

(n and r) and the output (${}^{n}C_{r}$).

1(b). Write a recursive C program to compute the value of ${}^{n}C_{r}$.

The value of n and r are read from keyboard. Print the result.

Hint: Try to write a recursive expression of ${}^{n}C_{r}$.

Assignment 2

2. Write a C function named Congruent() that takes three positive integers a, b and n. Assume, n is non-zero but a and b can be zero.

The function returns 1 if a (modulo n) = b (modulo n) and returns 0, if a (modulo n) ≠ b (modulo n)

Write a main program in C that reads an array of n number of positive integers including zero. Use the function Congruent() and find any 3 distinct pairs of integers (a, b) from the array in such that,

a (modulo n) = b (modulo n).

(a, b) and (c, d) are distinct pairs if a (modulo n) ≠ c (modulo n).

Contd.

Assignment 2 Contd.

Print the input array and the output i.e. the three distinct pairs.
(Choose n properly so that you can get at least 3 pairs)

Example

Input array: [29, 5, 23, 8, 40, 32, 16]

Output: (29, 8), (5,40), (23,16)

(since, 29 modulo 7 = 8 modulo 7 = 1;

5 modulo 7 = 40 modulo 7 = 5 and

23 modulo 7 = 16 modulo 7 = 2)

Assignment 3

3. Partsum of an integer n is defined as a way of writing n as a sum

1+1+1+1

1+1+2

1+3

2+2

4

Permuting the summands does not give a new partsum, e.g. 1+1+2 and 1+2+1 are treated as the same partsum.

Write a recursive C function to compute the number of partsums of n.

In order to avoid repetitions, choose the summands in a non-decreasing order. That is, if 1 and 2 are already chosen as the previous summands, the next summand cannot be less than 2.

contd.

Assignment 3 contd.

Pass two arguments to your recursive function. The first stands for the amount left to be balanced by summands and the second stands for the largest summand chosen so far.

Recursion stops when the first argument becomes 0.

Assignment 3 Contd.

Sample Output

Enter n: 5

Count of all partsums = 7

Enter n: 40
Count of all partsums = 37338

Enter n: 80
Count of all partsums = 15796476

Assignment 4 (Bonus)

Assignment 3 contd.

4(a). Repeat Assignment 3 with an additional constraint that no summand is repeated in the partsum. For example

For integer n = 4, the output will be 3 (instead of 5) and they are as follows

1 + 3

2 + 2

4

Modify the recursive function of Assignment 3 so that the output will be the count of partsums without repetitions.

contd.

Assignment 4 Contd.

4(b). Modify the recursive function of Assignment 3 so that the it counts the number of partsums having an odd number of summands. For example,

n = 4, there are two partsums with odd number of summands

1 + 1 + 2

4

n = 5, there are four partsums with odd number of summands

1 + 1 + 1 + 1 +1

1+ 1 + 3

1 + 2 + 2

5

[**Hint:** Pass a 3rd argument to the function in order to pass the information about how many summands have been chosen so far. When the balance (1st argument) becomes zero, whether an odd number of summands have been chosen so far.]

Thank You