**Department of Computer Science & Engineering**
Indian Institute of Technology Kharagpur

# Practice Sheet #06

Topic: Recursion in C

Date: 30-01-2017

1. What string does the following program print?

```
#include <stdio.h>
#include <string.h>

void strFunc2 (char A[], int n)
{
  char t;
  if (n <= 1) return;
     t = A[0]; A[0] = A[n-1]; A[n-1] = t;
     strFunc2(&A[1],n-2);
}

int main ()
{
   char A[10] = "PDS 2005";
   strFunc2(A,strlen(A));
   printf("%s", A);
}
```

   (a) 5002 SDP
   (b) 5DS 200P
   (c) PDS 2005
   (d) SDP 2005

2. In this exercise, we deal with complex numbers of the form $(\frac{a}{b}) + i(\frac{c}{d})$ with a, b, c, d integers such that b and d positive. The following structure represents such a complex number.

```
typedef struct {
     int a; /* Numerator of the real part */
     int b; /* Denominator of the real part */
     int c; /* Numerator of the imaginary part */
     int d; /* Denominator of the imaginary part */
} complexNumber;
```

Our aim is to compute the sum and product of two such complex numbers with each fraction reduced to the standard form (i.e., reduced to the form $\frac{m}{n}$ with gcd(m, n) = 1, n > 0).

Fill in the blanks.

```
int gcd(int a, int b) /*Compute the gcd of two integers*/
{
    if (a < 0) a = -a; if (b < 0) b = -b;
    if (a == 0) return b; if (b == 0) return a;
    return gcd(b,___________ );
}
```

```
void reduce(int *a, int *b) /* Reduce a fraction to lowest terms */
{
   int d;
   d = gcd(______,_______); *a =_______; *b = ______;
}

void sum(int a1, int b1, int a2, int b2, int *a, int *b)

/* (a1/b1)+(a2/b2) */
{
*a = a1*b2 + a2*b1; *b = b1*b2;
reduce( ___________, _________);
}

void mul(int a1, int b1, int a2, int b2, int *a, int *b)
/* (a1/b1)*(a2/b2) */
{
*a = a1 * a2; *b = b1 * b2;
reduce( __________, __________);
}

complexNumber compAdd(complexNumber z1, complexNumber z2)
/* Compute z1 + z2 */
{
complexNumber z;
                             /* Set the real part of z */
sum (_____________ );
                            /* Set the imaginary part of z */
sum (_____________ );
return z;
}

complexNumber compMul(complexNumber z1, complexNumber z2)
                /* Compute z1 * z2 */
{
/ *Some code for complex number multiplication */
}
```

3. What value does the call `h(5)` return, where `h` is defined as follows?

```
int h ( int n )
{
    if (n == 0) return 1;
    return 2*h(n-1);
}
```

(a) 25
(b) 32
(c) 64
(d) 120

4. Let the function `g` be defined as follows:

```
int g ( int n )
{
if (n < 2) return n;
return g(n/2);
}
```

What is the value returned by the call `g(142857)`?

(a) 0
(b) 1
(c) 2
(d) 71428

5. A two-dimensional character array is used to store a list of names. Each name is stored in a row. An empty string indicates the end of the list. Complete the ***recursive*** function `printNames()` to print the names stored in the two-dimensional array supplied as `p`.

```
void printNames (char (*p)[100])
{
______________________________
}

int main()
{
   char names[20][100] = {"Bombay", "Delhi", "Guwahati",
"Kanpur","Kharagpur", "Madras", "Roorkee", "" };
printNames(names);
return 0;
}
```

6. Write a *recursive* C function `findMed(…)`, which returns the median of a one-dimensional array of integers, that is, the element which is larger than half of the elements and smaller than half of the elements. Suppose $a_1 \leq a_2 \leq \cdot \cdot \cdot \leq a_n$ is the sorted version of the input array A. If n = 2k + 1, then the element $a_{k+1}$ is the median of A. On the other hand, if n = 2k, we take $a_k$ as the median of A. Your function should use a partitioning technique as in Quick sort. Use the following function prototype:

```
int findMed (int A[], int startidx, int endidx, const int
medidx);
```

Here, the second and the third parameters are the start and end indices of the current partition; the fourth parameter is the index of the median in the sorted version of A and is kept constant across the calls.

7. Write a recursive C function `char *lastOccur (char *s, char c)` which takes a string `s` and a character `c` as input; it returns a pointer to the last occurrence of `c` in `s`, or `NULL` if the character `c` does not occur in `s`. *Do not use any string library functions.*

8. Consider the following recursive C function.

```
unsigned int f(unsigned int n)
{
   if (n < 10) printf("%d",n);
   else {
      printf("%d", n%10);
      f(n/10);
      printf("%d", n%10);
   }
}
```

What does the call `f(351274)` print?

9. Consider the following piece of C code.

```
int S(int n, int k)
{
   if (k > n) return 0;
   if ((k == 1) || (k == n)) return 1;
   return S(n-1,k-1) + k * S(n-1,k);
}
```

(a) How many times is `S(…)` called (including the outermost call) to compute `S(5,3)`?

(b) What is the value returned by `S(5,3)`? Show your calculations.

(c) Write a recursive function `SMul()` to count the number of multiplications in the call `S(n,k)`.

10. Complete the following recursive function that returns the floating point value of a continued fraction $<a_0, a_1, . . . , a_{n-1}>$.
Note that $<a_i, a_{i+1}, . . . , a_{n-1}> = a_i + 1/(<a_{i+1}, a_{i+2}, . . . , a_{n-1}>)$ for $i = 0, 1, . . . , n - 2$.

```
double cfracrec ( int i, int n)
{
   int a;
   printf("Enter a_%d: ", i); scanf("%d", &a);
                  /* Read ai in a */

   if ( i == ________) return__________ ;
             /* The terminating case, no recursive call */

   return_________ ;
/* Make a recursive call and return */
}
```

The outermost call for computing $<a_0, a_1, . . . , a_{n-1}>$ should be:
`cfracrec( ___,___ )`

11. Consider the following piece of C program.

```
#include <stdio.h>

int F[10];

int fib(int n)
{
   printf("*");
   if (n <= 1) return 1;
   if (F[n] != 0) return F[n];
   F[n]=(fib(n-1) + fib(n-2));
   return F[n] ;
}

int main( )
{
   int j;
   for (j=0; j< 10; j++) F[j] = 0;
   fib(5);
}
```

How many `*s` will be printed by the following program?

12. For an integer *k* we have $a^{2k} = (a^k)^2$, and $a^{2k+1} = (a^k)^2 * a$.

    Use this observation to write a **recursive** function that, given a positive real number *a* and a non-negative integer *n*, computes $a^n$.
    The function should be efficient in terms of the total number of multiplications required to find the answer.

13. The following incomplete code is for a **recursive function** RecursiveArraySum, which calculates the sum of all the n-elements of an array passed as function argument.
    Write the missing pieces of the code such that the sum is calculated by recursively calling the function. The function RecursiveArraySum is invoked with a 4-element array of [5, 10, 20, 40] as argument. The base address of the array is at 10,000 (decimal). At every recursive call, write the values of the parameters passed and the return value from the function.

```
int RecursiveArraySum(int *arr, int n){
   if . . . . . . . . . return . . . . . . ;
   . . . . RecursiveArraySum(……………);
}
```

Ans.
if (n == 0) return 0;
else
return (a[0] + RecursiveArraySum(a+1,n-1));
OR
return (a[n-1] + RecursiveArraySum(a,n-1));

14. The following function is expected to compute the factorial of a positive number **n**. Is the given code error free? If not, then using one sentence state the reason why it will not work and also give the corrected code.

```
long fact(int n)
{
   long k=1;
   do
      k*=fact(i--);
   while(n>0);
   return k;
}
```

Ans. No, it uses both recursion and iteration.
long fact(int n)
{
long k=1;
for(i=0;i<n;i++)
k*=i;
return k;
}
Or
long fact(int n)
{
if (n<=1) return 1;
return n*fact(n-1);
}

15. What value will the following function return when called as `recur(3)`?

```
int recur(int data)
{
   int k;

   k=(data>2)?(recur(data-1)-recur(data-2)):1;
   return k;
}
```

16. Consider the following recursive function:

```
unsigned int f (unsigned int n)
{
    if (n <= 2) return 1;
    return f(n-3) + f(n-1);
}
```

What is the maximum height to which the recursion stack grows when the outermost call is `f(10)`? Assume that the stack is empty just before this outermost call.

(a) 5

(b) 9
(c) 13
(d) 32

17. Mathematically express the return value f (n) of the function supplied below. Your mathematical expression for f (n) must hold for all integers n > 0.

```
unsigned int f(unsigned int n)
{
if (n == 0) return 0;
return 2 * f(n - 1) + 1;
}
```

18. The Fibonacci sequence is defined as

```
F(0) = 0,
F(1) = 1,
F(n) = F(n-2)+F(n-1) for n > 2.
```

The following function defines another sequence G(n) for n = 0, 1, 2, 3, ….. .
How is the sequence G(n) mathematically related to the Fibonacci sequence F(n)? (Express G(n) in terms of F(n). Your expression should hold for all integers n > 0.)

```
int G (unsigned int n)
{
if (n == 0) return 0;
if (n == 1) return 1;
return G(n-2) - G(n-1);
}
```

19. Convert the recursive program in the left box to an iterative program in the right box.

```
int sum(int n)
{
   if (n < 1) return 0;
   return sum(n - 1) + sum(n - 2) + n;
}
```

20. What is printed by the following program?

```
#include<stdio.h>

void t(int n, char fp, char tp, char ap)
{
   if(n==1){
      printf(“%c%c”,fp,tp);
      return;
   }
   t(n-1,fp,ap,tp);
   printf("%c%c",fp,tp);
   t(n-1,ap,tp,fp); return;
}

void main()
{
 t(2,'x','y','z');
}
```

21. If our universe consists of only the set of nonnegative integers, the even and odd numbers can be characterized as follows: a number is *even* if its predecessor is odd, a number is *odd* if is not even, the number 0 is even by definition. Complete the C functions below which return 1 when the input number *n* is even or odd respectively. Both the functions are defined in the same program and they call each other.

```
int IsEven(unsigned int n)
{
   if (n == 0) {
      return 1;
   }
   else {
      return_________;
   }
}

int IsOdd(unsigned int n)
{
   return__________;
}
```

22. A recursive algorithm may require more computation time and memory than its iterative version.

    (a) TRUE
    (b) FALSE

23. Let us assume xn is defined as follows:

    $$xn = x^{n/2} * x^{n/2} \text{ if n is even}$$
    $$= x * x^{n/2} * x^{n/2} \text{ if n is odd}$$

    (a) Write a recursive C function to compute xn based on the above definition. Clearly mention suitable base cases.
    (b) Find out how many function calls are made to compute x16 using your function.

24. Consider the following C function:

```
int f(int n)
{
   static int i = 1;
   if (n >= 5)
```

```
        return n;
    n = n+i;
    i++;
    return f(n);
}
```

The value returned by `f(1)` is

(a) 5
(b) 6
(c) 7
(d) 8

25. Consider the following C-program:

```
void foo(int n, int sum)
{
  int k = 0, j = 0;
  if (n == 0) return;
    k = n % 10;
  j = n / 10;
  sum = sum + k;
  foo (j, sum);
  printf ("%d,", k);
}

int main ()
{
  int a = 2048, sum = 0;
  foo (a, sum);
  printf ("%d\n", sum);

  getchar();
}
```

What does the above program print?

(a) 8, 4, 0, 2, 14
(b) 8, 4, 0, 2, 0
(c) 2, 0, 4, 8, 14
(d) 2, 0, 4, 8, 0

26. Consider the following C function, what is the output?

```
#include <stdio.h>
int f(int n)
{
    static int r = 0;
    if (n <= 0) return 1;
    if (n > 3)
    {
        r = n;
        return f(n-2)+2;
    }
    return f(n-1)+r;
}

int main()
{
    printf("%d", f(5));
}
```

(a) 5
(b) 7
(c) 9
(d) 18

27. Consider the following recursive C function that takes two arguments

```
unsigned int foo(unsigned int n, unsigned int r) {
  if (n  > 0) return (n%r +  foo (n/r, r ));
  else return 0;
}
```

What is the return value of the function foo when it is called as `foo(345, 10)` ?

(a) 345
(b) 12
(c) 5
(d) 3

28. Consider the same recursive C function that takes two arguments

```
unsigned int foo(unsigned int n, unsigned int r) {
  if (n  > 0) return (n%r +  foo (n/r, r ));
  else return 0;
}
```

What is the return value of the function foo when it is called as foo(513, 2)?

(a) 9
(b) 8
(c) 5
(d) 2

29. Consider the following function

```
int f(int j)
{
    static int i=50;
    int k;
    if(i==j)
    {
            printf("something");
            k=f(i);
            return 0;
        }
        else
        return 0;
    }
```

(a) Function will return 0 for all j
(b) Function prints "something " for all values of j
(c) For i==j function goes into infinite loop
(d) For i==j function returns 0

30. Consider the following function

```
double f(double x){
  if (abs(x*x - 3) < 0.01) return x;
  else return f(x/2 + 1.5/x);
}
```

Give a value q (to 2 decimals) such that f(q) will return q:_____.

31. Consider the following C function.

```
int fun (int n)
{
  int x=1, k;
  if (n==1) return x;
  for (k=1; k<n; ++k)
     x = x + fun(k) * fun(n – k);
  return x;
}
```

The return value of `fun(5)` is __________.

(a) 0
(b) 26
(c) 51
(d) 71

32. Consider the following recursive C function. If get(6) function is being called in main() then how many times will the get() function be invoked before returning to the main()?

```
void get (int n)
{
   if (n < 1) return;
   get(n-1);
   get(n-3);
   printf("%d", n);
}
```

(a) 15

(b) 25
(c) 35
(d) 45

## Problems for Programming Practice

(After the successful studies of Lecture 06 (Recursion in C), the students are supposed to solve the following problems in C programming language.)

1. What does the following program return?

```
a.  int foo5 ( unsigned int n )
     {
             if (n == 0) return 0;
             return 3*n*(n-1) + foo5(n-1) + 1;
    }

b.  int foo6 ( char A[] , unsigned int n )
    {
     int t;

    if (n == 0) return 0;
    t = foo6(A,n-1);
    if ( ((A[n-1]>='a') && (A[n-1]<='z')) ||
            ((A[n-1]>='A') && (A[n-1]<='Z')) ||
            ((A[n-1]>='0') && (A[n-1]<='9')) )
            ++t;
    return t;
 }

c.  int foo7 ( unsigned int a , unsigned int b )
    {
    if ((a == 0) || (b == 0)) return 0;
            return a * b / bar7(a,b);
   }

   int bar7 ( unsigned int a , unsigned int b )
    {
            if (b == 0) return a;
    return bar7(b,a%b);
  }
```

```
d.   int foo8 ( unsigned int n )
     [
            if (n == 0) return 0;
            if (n & 1) return -1;
            return 1 + bar8(n-1);
     }

 int bar8 ( int n )
 {
    if (!(n & 1)) return -2;
    return 2 + foo8(n-1);
 }
```

2. A rational number is defined by a pair of integers (a,b) with b > 0 and is interpreted to stand for a/b. A rational number a/b is said to be in the reduced form if gcd(a,b)=1.

a. Define a structure to represent a rational number.
b. Write a function that returns the rational number 0/1. This function can be used to initialize a rational number variable.
c. Write a function that, given a rational number, returns its reduced form.
d. Write a function that, upon input of two rational numbers, returns the sum of the two input numbers in the reduced form.
e. Repeat the previous part for subtraction, multiplication and division of rational numbers.

3. Consider the following set of real numbers:

```
A = { a + b sqrt(2) | a,b are integers }.
```

a. Define a structure to represent an element of A.
b. Write a function that, upon input of two elements of A, returns the sum of the input elements.
c. Repeat the last part for subtraction and multiplication of elements of A.
d. It is known that the element a + b sqrt(2) has an inverse in the set A if and only if $a^2 - 2b^2 = 1$ or -1. Write a function that, given an invertible element of A, returns the inverse of the element. If the input to the function is not invertible, the function should return 0 after prompting an error message.

4. A circle in the X-Y plane is specified by three real numbers a,b,c. The real numbers may be interpreted in two possible ways. The first possibility is that (a,b) represents the center and c the radius of the circle. In the second representation, we refer to the equation of the circle as:

$$X^2 + Y^2 + aX + bY + c = 0.$$

So a structure holding three double variables together with a flag indicating the particular interpretation suffices to store a circle.

a. Write a function that converts a circle structure from the first to the second representation.
b. Write a function that converts a circle structure from the second to the first representation.
c. Write a function that, upon input a circle and two real numbers x,y, checks whether the point (x,y) lies inside, on or outside the circle. Note that the input circle may be of any of the two representations.
d. Write a function that, upon input two circles each with any representation, determines whether the circles touch, intersect or do not intersect.
e. Write a function that, upon input a circle in any representation, returns the side of a square that has the same area as the input circle.

5. Write a recursive function that computes the binomial coefficient C(n,r) using the inductive definition:

$$C(n,r) = C(n-1,r) + C(n-1,r-1)$$

for suitable values of n and r. Supply appropriate boundary conditions.

6. Write a Function to recursively compute the harmonic mean of an array of numbers. In the main function, create an array of size 10. Input integers from the user till a negative number is given as input or the 10 elements have been filled up. Find the harmonic mean of the elements of this array.

7. Write a function to recursively compute the sum of digits of a positive integer. The function has to be recursive. Function Protype : int sum(int n);

8. A rectangle in the X-Y plane can be specified by eight real numbers representing the coordinates of its four corners. Define a structure to represent a rectangle using eight double variables. Notice that here we do not assume the sides of a rectangle to be necessarily parallel to the X and Y axes. Notice also that by a rectangle we mean only the boundary (not including the region inside).

   a. Write a function that, upon input a structure of the above kind, determines whether the structure represents a valid rectangle.
   b. Write a function that, upon input a valid rectangle, determines the area of the rectangle.
   c. Write a function that, upon input a valid rectangle and two real numbers x, y, determines whether the point (x,y) lies inside, on or outside the rectangle.

9. Define a structure customer to specify data of customer in a bank. The data to be stored is: *Account number* (integer), *Name* (character string having at most 50 characters), and *Balance* in account (integer).

Assume data for all the 10 customers of the bank are stored in the structure array :

```
struct customer bank[10];
```

The function, transaction, is used to perform a customer request for withdrawal or deposit to her account. Every such request is represented by the following three quantities: *Account number of the customer*, *request type* (0 for deposit and 1 for withdrawal) and *amount*. The function prototype is as follows:

```
int transaction ( int account_number, int request_type, int
amount,    struct customer bank [ 10 ] )
```

The transaction function returns 0 if the transaction fails and 1 otherwise. The transaction fails only when the account balance is less than the withdrawal amount requested.

The array *bank* (defined above) is another input to the transaction function and is suitably updated after every request. In case of a failed transaction no change is made in the bank array.

Write a main() function which populates the array *bank* with values for 5 customers. Also, the main() should take a withdrawal request from the user (i.e., read values for account number, amount), and call the *transaction* function, and thereby print if it is a valid transaction. If valid, it should print the balance after the withdrawal.

10. n-th (n ≥ 2) Fibonacci number is defined as follow;

$$F_n = F_{n-1} + F_{n-2} \text{ with } F_1 = F_0 = 1$$

    Write a recursive function *int Fibonacci(int n)* , which would return the n-th Fibonacci number when you call it from main for an input *n* .

11. Read any two positive numbers *a* and *b*. Write a recursive function *int gcd(int a, int b)*, which would return the greatest common divisor of *a* and *b*. You should read the numbers *a* and *b* in the main function and call *gcd(a,b)* in the main.

12. The n-th Harmonic number is defined as $H_n = 1 + \frac{1}{2} + \frac{1}{3} + \frac{1}{4} + \cdots$. Write a recursive function to calculate $H_n$, for any integer $n > 0$.

--*--