1. [10] Write three versions of the function that computes \( f \) defined as follows: \( f(i) = f(i-1) + f(i-2) + f(i-3) \), and \( f(0) = 0, f(1) = 1, f(2) = 2 \), for \( i \geq 0 \),
   (1) purely recursive
   (2) iterative
   (3) a modified recursive function using recursion and an array to avoid redundant computation.

2. [15] Given a set \( I \) of \( n \) 16-bit ranges denoted by \([b_i, e_i]\) for \( i = 0 \) to \( n - 1 \), where \( 0 \leq b_i \leq 65535 \) and \( 0 \leq e_i \leq 65535 \).
   (a) Write a function that takes this set of ranges as the first parameter \( (I) \) and a 16-bit number \( v \) as the second parameter and the third parameter is the set of ranges \( (R) \) that covers \( v \). For example, if the set of input ranges is \( I = \{ [3, 19], [11, 33], [18, 80], [80, 100] \} \) and \( v = 18 \), then \( R = \{ [3, 19], [11, 33], [18, 80] \} \).
   (b) Write another function that takes this set of ranges as the first parameter \( (I) \) for input and a set of 16-bit numbers \( (p_i) \) for \( i = 0 \) to \( k - 1 \) as the second parameter for output so that \( T \) is the union of \( b_i - 1 \) and \( e_i \) for \( i = 0 \) to \( n - 1 \). For example, if the set of input ranges is \( I = \{ [3, 19], [11, 33], [18, 80], [80, 100] \} \), then \( T = \{ 2, 10, 17, 19, 33, 79, 80, 100 \} \).

3. [30] Assume a binary tree of integers is stored in array int \( a[SIZE] \) and \( SIZE = 2^v \) as follows. The tree size \( (2^v - 1) \) is stored in element \( a[0] \), the integer in root is stored in \( a[1] \), and the integers in \( a[i] \)'s two children are stored in \( a[2^i] \) and \( a[2^i + 1] \) for \( i = 1 \) to \( 2^{v-1} - 1 \). For example, when we declare an array int \( a[SIZE] = \{ 15,1,2,3,4,5,6,7,8,9,10,11,12,13,14,15 \} \), it means we have a binary tree illustrated as

```
      1
     /\  
    2 3 /\  
   /\ 6/\ 7 /\  
  4/\ 5/\ 8    /\  
 /\ /\ /\  
8 9 10 11 12 13 14 15
```

Before mirror
mirror a subtree rooted at node \( i = \) mirror the subtree rooted at node \( 2i \) and
mirror the subtree rooted at node \( 2i + 1 \)

(a) Write a recursive function that can mirror a binary tree.
(b) Write an iterative function that can mirror a binary tree.
Please define the function prototype and describe its parameters.

4. [15] Our Internet IP address is 32 bits, which is usually represented as four 8-bit numbers separated by dot signed (129.160.96.1). Please a program based on command line arguments to do the following:
   (a) Input an IP address in the format of four numbers and three dot signs and store it in an unsigned variable and then print out its 32-bit bit pattern.
   (b) Input a 32-bit bit pattern and print out it as an IP address in the dot format.

5. [15] Please trace the following program and show what gets printed? Explain. Also, modify this program to write a recursive program int combinations(A, n, k) that you can print out all the combinations of \( k \) numbers out of \( n \) different numbers stored in an array \( A \) with additional rules: (1) the order of \( A[0], A[1], ..., A[n-1] \) must remain and (2) the sequence of these \( k \) numbers must be in an increasing order. For example, assume there are 4 numbers 4, 1, 2, 3 stored in array int \( A[4] \). Calling this recursive function combinations(A, 4, 2) will return a count 3 and print out (1, 2), (1, 3),
and (2, 3), or calling combinations(A, 4, 3) will return a count 1 and print out (1, 2, 3). Your recursive program must consider to avoid the unnecessary recursive function calls.

```c
#include <stdio.h>
define N 4
int boolfunc(int *var, int m);
int recursivebool(int *var, int n);
main()
{
    int varbool[20];
    recursivebool(varbool, N);
}
int boolfunc(int *var, int m)
{
    int result=var[0], i;
    for (i=1; i<m; i++) result = (result && var[i]);
    return result;
}
int recursivebool(int *var, int n)
{
    int localvar[20], i, j;
    if (n == 0){
        for(i=0; i<N; i++) printf("%d ", var[i]);
        print("%d\n", boolfunc(var, N));
        return;
    }
    for (j=0; j<1; j++) { 
        var[n-1] = j;
        recursivebool(var, n - 1);
    }
}

int determinant(int f[][10], int x)
{
    1. int pr=1, c[10], d=0, b[10][10], j, p, q, t;
    2. if(x==2) return (f[1][1]*f[2][2] - f[1][2]*f[2][1]);
    3. for(j=1; j<=x; j++){
    4.      int r=1,s=1;
    5.      for (p=1; p<=x; p++) {
    6.          for (q=1; q<=x; q++) {
    7.              if (p!=1 && q!=j) {
    8.                  b[r][s]=f[p][q];
    9.              s++;
    10.             if(s > x-1) { r++; s=1; }
    11.            }
    12.          }
    13.      }
    14.      for(t=1,pr=1;t<=(1+j);t++) pr=(-1)^pr;
    15.      c[j] = pr*determinant(b,x-1);
    16.  }
    17.  for(j=1,d=0;j<=x;j++) d += (f[1][j]*c[j]);
    18.  return(d);
}
```

6. [15] Please trace the following program and explain what it is doing. Explain the idea of this function and show an example. Can you simplify line 14?

```c
#include <stdio.h>
define N 4
int boolfunc(int *var, int m);
int recursivebool(varbool, N);
main()
{
    int varbool[20], i, j;
    if (n == 0){
        for(i=0; i<N; i++) printf("%d ", var[i]);
        print("%d\n", boolfunc(var, N));
        return;
    }
    for (j=0; j<1; j++) { 
        var[n-1] = j;
        recursivebool(var, n - 1);
    }
}

int determinant(int f[][10], int x)
{
    1. int pr=1, c[10], d=0, b[10][10], j, p, q, t;
    2. if(x==2) return (f[1][1]*f[2][2] - f[1][2]*f[2][1]);
    3. for(j=1; j<=x; j++){
    4.      int r=1,s=1;
    5.      for (p=1; p<=x; p++) {
    6.          for (q=1; q<=x; q++) {
    7.              if (p!=1 && q!=j) {
    8.                  b[r][s]=f[p][q];
    9.              s++;
    10.             if(s > x-1) { r++; s=1; }
    11.            }
    12.          }
    13.      }
    14.      for(t=1,pr=1;t<=(1+j);t++) pr=(-1)^pr;
    15.      c[j] = pr*determinant(b,x-1);
    16.  }
    17.  for(j=1,d=0;j<=x;j++) d += (f[1][j]*c[j]);
    18.  return(d);
}
```