IS2610 data structure
Hw2 Solutions
By Kaihong Liu

//IS2610 HW2 Kaihong Liu Q1

//COMPLEX.h

```
typedef struct {float Re; float Im;} Complex;
Complex COMPLEXinit(float, float);
        float Re(Complex);
        float Im(Complex);
Complex COMPLEXmult(Complex, Complex);
Complex COMPLEXadd(Complex, Complex);
Complex COMPLEXsub(Complex, Complex);
int COMPLEXequal(Complex, Complex);
```

//client.c

```
#include <stdio.h>
#include <stdlib.h>
#include <math.h>
#include "COMPLEX.h"
#define PI 3.141592625

main()
{
        float i,j,m,n;
        int k;
        char c;
        Complex t, x,y;
        printf("Enter the first float number  ");
        scanf("%f", &i);
        printf("Enter the second float number  ");
        scanf("%f", &j);
        t=COMPLEXinit(i,j);
        printf("First Complex Number: %6.3f %6.3f \n", Re(t), Im(t));
        printf("Enter the third float number  ");
        scanf("%f", &m);
        printf("Enter the fouth float number  ");
        scanf("%f", &n);
        x=COMPLEXinit(m,n);
        printf("Seond Complex Number: %6.3f %6.3f \n\n", Re(x), Im(x));

        printf("Please select the operation by input the option number (1, 2, 3)\n");
```

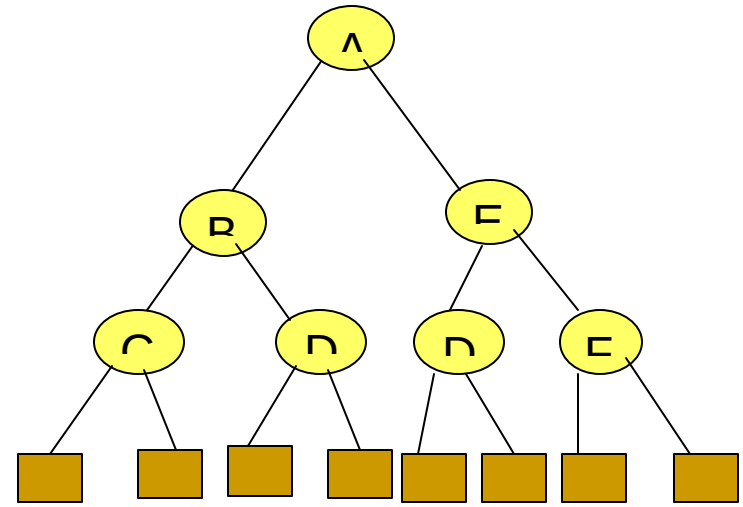
```
	printf("1. Add two complex numbers.\n2. Substract two complex numbers.\n3. Multiply two complex numbers.\n4. Compare two complex numbers\n5. Quit\n");

	scanf("%d",&k);

	while (k>0 && k<5)
	{
		if (k==1)
		{	y=COMPLEXadd(t,x);
			printf("Complex Number after add operation: %6.3f %6.3f \n", Re(y), Im(y));
		}
		else if (k==2)
		{
			y=COMPLEXsub(t,x);
			printf("Complex Number after sub operation: %6.3f %6.3f \n", Re(y), Im(y));
		}
		else if (k==3)
		{
			y=COMPLEXmult(t,x);
			printf("Complex Number after multply operation: %6.3f %6.3f \n", Re(y), Im(y));
		}
		else if (k==4)
		{
			k=COMPLEXequal(t,x);
			if (k==1)
			{printf("Two Complex Numbers are equal.\n");}
			else {printf("Two Complex Numbers are not equal.\n");}
		}
		printf("Do you want to do another operation by press y or n?\n");
		scanf("%s", &c);
		if (c=='y' || c=='Y')
		{
			printf("1. Add two complex numbers.\n2. Substract two complex numbers.\n3. Multiply two complex numbers.\n4. Quit\n");

			scanf("%d",&k);
		}

		else
		{ printf("Bye....\n");
			break;
		}
	}

}
```

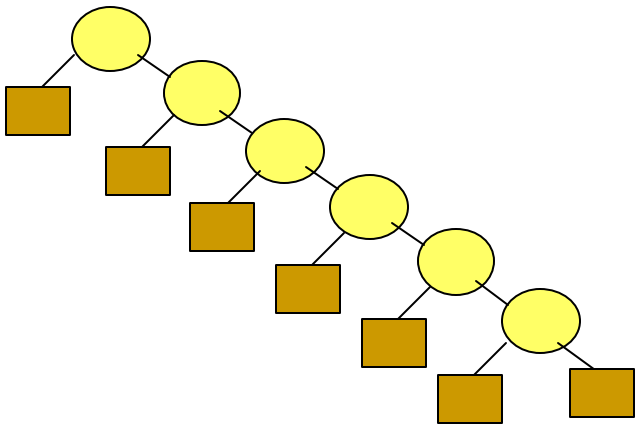
```
/*  Complex.c*/

#include "COMPLEX.h"

Complex COMPLEXinit(float Re, float Im)
{ Complex t; t.Re=Re; t.Im=Im; return t;}

float Re(Complex z)
{ return z.Re;}

float Im(Complex z)
{ return z.Im;}

Complex COMPLEXmult(Complex a, Complex b)
{
        Complex t;
        t.Re = a.Re*b.Re - a.Im*b.Im;
        t.Im = a.Re*b.Im + a.Im*b.Re;
        return t;
}

Complex COMPLEXadd(Complex a, Complex b)
{
        Complex t;
        t.Re = a.Re+b.Re;
        t.Im = a.Im+b.Im;
        return t;
}

Complex COMPLEXsub(Complex a, Complex b)
{
        Complex t;
        t.Re = a.Re-b.Re;
        t.Im = a.Im-b.Im;
        return t;
}

int COMPLEXequal(Complex a, Complex b)
{
        if ((Re(a)==Re(b)) && (Im(a)==Im(b)))
                return 1;
        else return 0;

}
```

```
//IS2610 hw2 kaihong Liu Q2
#include <stdio.h>
#include <stdlib.h>

float DynamicCN(int N);
float RecursiveCN(int N);

main ()
{
 int N;
 float a,b;

 printf("Please enter the number of N: ");
 scanf("%d", &N);
 a=DynamicCN(N);
 printf("Here is Dynamic claculated result %6.3f: \n",a);

 b=RecursiveCN(N);
 printf("Here is Recursive claculated result %6.3f: \n",b);

}


float DynamicCN(int N)
{
        int i=0,k;
        float x=0.0;
        float *C=malloc((N+1)*(sizeof C));
        C[0]=1; //we know C[0]=1

        //printf("Here is C[0]: %6.3f \n", C[0]);
        for (i=1; i<=N; i++)
        {
                x=0.0;
                for (k=1; k<=i; k++)
                {
                        x=x+C[k-1]+C[i-k];
                }
                C[i]=i+ (x/i);
                printf("Here is C[%d]: %6.3f\n", i,C[i]);
        }
        return C[N];
}

float RecursiveCN(int N)
{
        int i;
```

```
        float x=0.00;
        if (N == 0) return 1;

        for (i=1; i<=N; i++)
                {

                        x=x+RecursiveCN(i-1)+RecursiveCN(N-i);

                }

                return N+(1.0/N)*x;

}
```

5.71





Give upper and lower bounds on the number of leaves in a binary tree with N nodes.
Lower bound is 1,

If the tree looks like this

There would be only one leaf node.
But if the tree is complete balanced tree

If N is the internal node number. Then the external node number would be N+1. The leaf would be one level up above external node and since each node has 2 external node maximum. The leaf number would be (N+1)/2 if N+1 is odd number need to take floor of (N+1)/2 => $\lfloor (N+1)/2 \rfloor$ This is the upper bound.

If N is the number of total node which include both internal and external nodes. Then let n represent internal node number the external node number will be n+1
N=n+n+1=2n+1 n=(N-1)/2
External node number=(N-1)/2 + 1=(N+1)/2
Maximum leaf number = (N+1)/2/2=(N+1)/4. Take floor of this number $\lfloor (N+1)/4 \rfloor$
This is the upper bound if N is the number of total node.

5.78
A complete binary tree is one with all levels filled, except possible the final one, which is filled from left to right, as illustrated in Fig 5.24. Prove that the internal path length of a complete tree with N nodes is between NlgN and NlgN+N.

This question itself has problem.