June 6, 2013

DATA ANALYSIS RESIT EXAM

Q1) A binary search tree has the nodes that contains integer values. Write C function to delete the second largest element of the tree. The root pointer is known.

Q2) Explaning the C program given on the back page, determine the printed values.

Q3)

60, 120, 48, 12, 210, 83, 10, 30, 400, 38, 5, 8, 70,26, 500,600, 180 keys are given

a) Using first 12 keys, constract reordered Hash table. And explain the avantages of the reordered hash table. **h(k)= k% table_size.**

b) Using all keys, respectively, constract a 5th-order B-Tree. Delete the keys 60 and 10.

Q4)

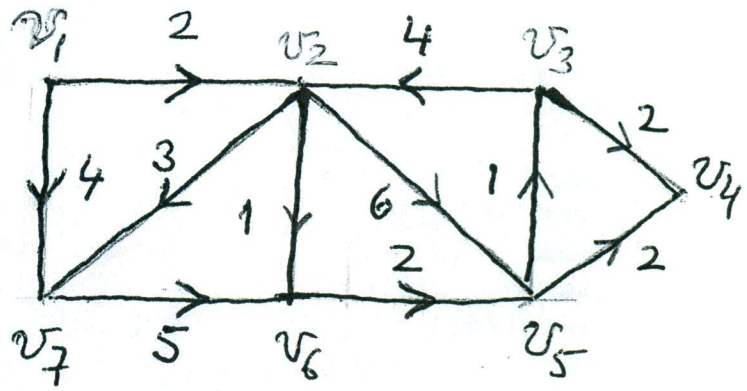
For a binary tree with **n** nodes, the number of successful comparision is given by $C_n = (2I+n)/n$. Where **I** is the internal path length of the tree. Show that $C_n = 2\lg(n) - 3$ in the best case and $C_n = n$ in the worst case.

Q5)

a) Adjacency matrix of a garph **G** is given. Draw the linked list representation of the same graf **G**.

b) Find the path with the smallest length from **v1** to other nodes in graph **G1** by using Dijkstra algorithm..

$$A = \begin{array}{c|cccccc} & A & B & C & D & E & F \\ A & 0 & 1 & 0 & 1 & 0 & 0 \\ B & 0 & 0 & 1 & 0 & 1 & 1 \\ C & 0 & 0 & 0 & 0 & 0 & 1 \\ D & 0 & 1 & 0 & 0 & 1 & 0 \\ E & 0 & 0 & 0 & 0 & 0 & 1 \\ F & 0 & 0 & 0 & 0 & 0 & 0 \end{array}$$



```
#include<stdio.h>
#include<conio.h>
typedef struct ope{
 char op;
 struct ope *next;
 }OPE;
typedef struct say{
 int sa;
 struct say *next;
 }SAY;
 OPE *on1; SAY *on2;
 main()
 {
   OPE*olustur_1( char [] ) ;
   SAY*olustur_2( int[] );
   OPE* pop1( void );
   SAY*pop2( void );
   void push( int );
 char dizi_1 [ 5 ] = {'+' ,'/' , '*', '-', '*' };
   char is;
   int s1, s2;
   int dizi_2 [ 6 ]={5, 3, 2, 6, 2, 2 };
    on1 = olustur_1( dizi_1 );
   on2 = olustur_2( dizi_2 );
   while( on1 != NULL)
    {
     is = pop1() -> op;
     s1 = pop2() -> sa;
     s2 = pop2() -> sa;
    switch( is )
    {
      case '+':
        push( s1 + s2 );
      break;

      case  '-':
        push( s1 - s2 );
      break;

      case '*':
        push( s1 * s2 );
      break;
      case '/':
     push( s1 / s2 );
      break;
    }
 }
   printf("\n s2=%d",on2->sa);
}
```

```
 OPE *olustur_1( char dizi_1[] )
 {
   OPE *ptr, *pp;
   int i, n = 5;
 pp = ptr = (OPE *)malloc( sizeof( OPE ) );

   for( i = 0; i <= n - 2; i++ )
    {  ptr -> op = dizi_1[ i ];
     ptr=ptr->nexti=
(OPE*)malloc(sizeof(OPE));
    }
     ptr -> op = dizi_1[ n - 1 ];
     ptr -> nexti = NULL;
     return ( pp );
 }//end of olustur_1
////////////////////////////////////////
SAY *olustur_2( int dizi_2[] )
 {  SAY *ptr, *pp;
   int i, n = 6;
   pp = ptr = (SAY )malloc(sizeof(SAY));

   for( i = 0; i <= n - 2; i++ )
    { ptr -> sa = dizi_2[ i ];
     ptr=ptr->next
=(SAY*)malloc(sizeof(SAY));}
     ptr -> sa = dizi_2[n-1];
     ptr -> next = NULL;
     return( pp );
   }//end of olustur_2
OPE *pop1()
 {  OPE *ptr;
  ptr = on1;
  on1 = on1 -> next;
  return( ptr );
 }
SAY *pop2()
 {  SAY  *ptr;
  ptr = on2;
  on2 = on2 -> next;
  return( ptr );
 }
/**********************/
void push( int ss )
{  SAY *ptr;
  ptr = (SAY *)malloc(sizeof(SAY));
  ptr -> sa = ss;
  ptr -> next = on2;
  on2 = ptr;
}
```