May 24 2013

DATA ANALYSIS FINAL EXAM

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

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

Q3)

Using the keys values respectively
48, 30, 100, 75, 60, 65, 40, 15, 35, 55

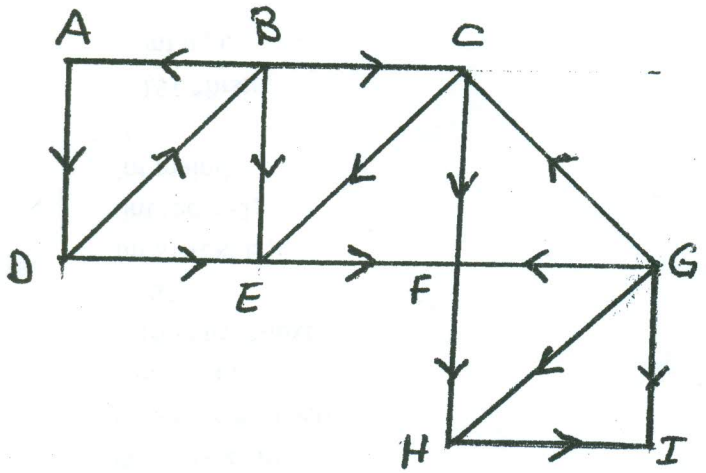
a) Constract heap and delete the root from the heap two times,
b) Form reordered Hash table (h(k) = k% 10, use the linear probing).
c) Show that an AVL tree of height **h** has at least **f(h+3)-1** nodes, where **f(i)** is the *ith* Fibonacci number.

Q4)

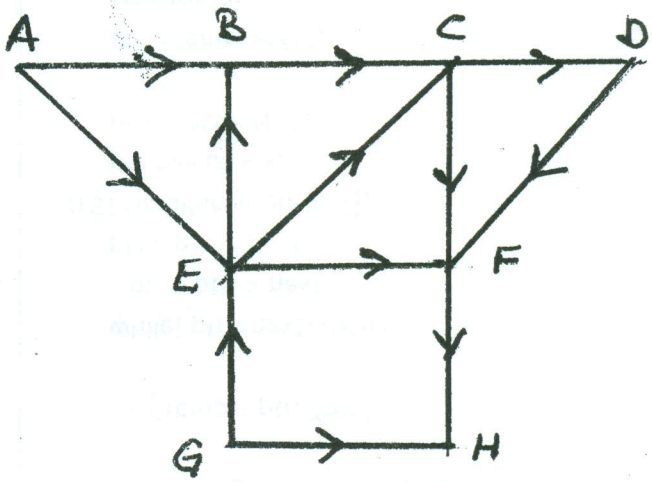
a) Constract a 3th order B-tree using following keys, respectively
50, 86,15,180,67,65,38,6,11,46,70,200,120,80
b) Explain the searching operation for key 88 in this tree,
c) Delete the keys 15, 86, 120 from the tree, respectively.

Q5)

a) Make the breadth first and depth first traversal of the Graph-1., starting node A.
b) Obtain a topological sorting for Graph-2.



Graph - I



Graph-II.

```
#include <stdio.h>
#include <conio.h>
typedef struct list{
 int value;
 struct list *next;
 }LST;
int STACK[ 6 ];
 int top = -1;
int main()
 {
   LST *first;
   int m, i;

   LST* function_A (int, LST*);
   LST* function_B (int, LST*);
   void push(int);
   int pop();
   printf("\n Numaranızın son rakamı..?");
   scanf("%d", &m);
   int A[6] = { 7, 2, m,3, 8, 6}, CC;
   for( i = 0; i<=5;i++)
    push(A[i]);

   first = (LST*)malloc(sizeof(LST));
   first->value = pop();
   first->next = NULL;


   while(top!=-1)
    {
      CC = pop();
      if( first -> value > CC )
        first = function_A( CC, first);
      else
        first = function_B( CC, first);
     }
   while(first != NULL)
    {
      printf(" %d ",first->value);
      first = first -> next;
     }

  }
```

```
////////////////////////////////
LST* function_A( int x, LST*ptr)
 {
    LST*temp = ptr, *ptr1;

    while( ptr->next!=NULL)
      ptr = ptr -> next;
    ptr = ptr -> next =
(LST*)malloc(sizeof(LST));
    ptr->value = x;
    ptr->next = NULL;

    ptr = temp->next;
    free(temp);
    return(ptr);
}
////////////////////////////////////
LST* function_B(int x, LST* first)
 {
   LST*ptr;
   ptr = (LST*)malloc(sizeof(LST));
   ptr->value=x;
   ptr->next = first->next;
   first->next = ptr;
   return(first);
 }
////////////////////////////////
void push(int CC)
{
   STACK[ ++top ] = CC;
}
//////////////////////
int pop()
{
   return(STACK [ top -- ]) ;
}
```