CEC 14th - Past Year Paper Solution 2013-2014
Sem2 CE1007/CZ1007 – Data Structures

Solver: Chandra Bhavya
Email address: chan0727@e.ntu.edu.sg
Feel free to email me if there is any error in the solutions.

1.a)

n=-2,*r=-240
n=2,*r=-240
n=6,*r=-240
n=10,*r=-240
m=10,r=-240

b)

```
int readin(PhoneBk *p){
    int size=0;
//Enter initial name
    printf("enter END to stop entering data.\nEnter name:");
    scanf("%s", p[size]->name);
    while(strcmp(p[size]->name,"END")!=0)
    {
       fflush(stdin);
        printf("\nEnter telephone:");
        scanf("%s,",p[size]->telno);
        size++;
        //Size is always index+1, so for index 0, size is 1
        fflush(stdin);
        printf("Enter name:");
        scanf("%s,",p[size]->name);
//if user enter END here the size is not increased because while doesn't
execute.
    }

    return size;
}
void search(PhoneBk *p, int size, char *target){
    int i;
    for(i=0;i<size;i++)
    {
        if(strcmp(p[i].name,target)==0)
        {
            printf("The telephone number for %s is %s \n", p[i]->name,
p[i]->telno);
            return;
        }
    }
    printf("ERROR SEARCH");
}
```

c) /*missing code segment */

```
  p2=str;
    //Dynamically allocated memory
    p1=malloc(n);
    *(p1+n)='\0';
    //Reverses and stores string str in pt1
    while(*p2!='\0'){
        *(p1+n-1)=*p2;
        p2++;m
        n--;
    }
    if(strcmp(str,p1)==0)
        printf("PALLINDROME");
    else
        printf("NOT PALLLINDROME");
    free(p1);
```

2. a)

4,4
3,6,9,12,15,
9,6

b)

```
void max(int m[SIZE],int *max1, int *max2)
{
    int i;
//Assumes that max1 is the number at 1st position and max2 is number at
2nd position
    *max1=m[0]>m[1]?m[0]:m[1];
    *max2=m[0]>m[1]?m[1]:m[0];
    for(i=2;i<SIZE;i++){ //loop goes from the 2nd position to last
position because we checked 1& 2
        if(*max1<m[i]){
            *max2=*max1;              //Changes values of max2 to present
max1
            *max1=m[i];
            continue;}
        if(*max2<m[i]){
            *max2=m[i];}
    }}
```

c)

For (i,j) As we can see for the Right Diagonal the elements occurs when i==j
For the left diagonal elements occur when i+j=2

0,0	0,1	0,2
1,0	1,1	1,2
2,0	2,1	2,2

```
#include <stdio.h>
#define SIZE 9
void max(int m[SIZE], int *max1, int *max2);

int main()
{
    int a[3][3]={1,2,3,4,5,6,7,8,9};
    int i,j,sum1=0,sum2=0;
    for(i=0;i<3;i++)


        for(j=0;j<3;j++){
            if(i==j)
                sum1+=a[i][j];
            if(i+j==2)
                sum2+=a[i][j];
        }

    printf("sum1=%d,sum2=%d\n",sum1,sum2);
}
```

3.a)

Stack & Queue are a special case of Linked List i.e. they are limited access sequential data structures.

A stack is based on the LIFO technique i.e. it can be only accessed from the top

Queue is based on FIFO technique i.e. it can be accessed from both ends but only addition occurs at the end and deletion from front.

Linked List are simple sequential access, and an item may be removed or added in any index position.

b)

STACK	QUEUE
Based on LIFO	Based on FIFO
pop and push from top only	pop() occurs from front, push() occurs from back
useful when the last item requires to be removed first	useful in case application need priority based on first come first serve basis

c)

NOTE: The definition of Double Linked List is assumed same as lecture notes.

```
int repairDblLinkedList(DblLinkedList *ll)
{
    ListNode *temp=ll->head;
    int broken=0;
    if(temp==NULL)
        return 0;
    //Begin by checking the second node
    temp=temp->next;

    while(temp!=NULL){

        if(temp->prev!=NULL)
{
        temp->prev=NULL;
            broken++;
}
        temp=temp->next;
    }
    return broken;
}
```

d)

```
typedef struct_coeff{
float coeff;
int exponent;
struct node *next
}PolynNode;
```

The PolyNode will store the coeff (a) and the power of x. Various operations can then be performed such as calculating roots etc by utilising coeff and power of x.

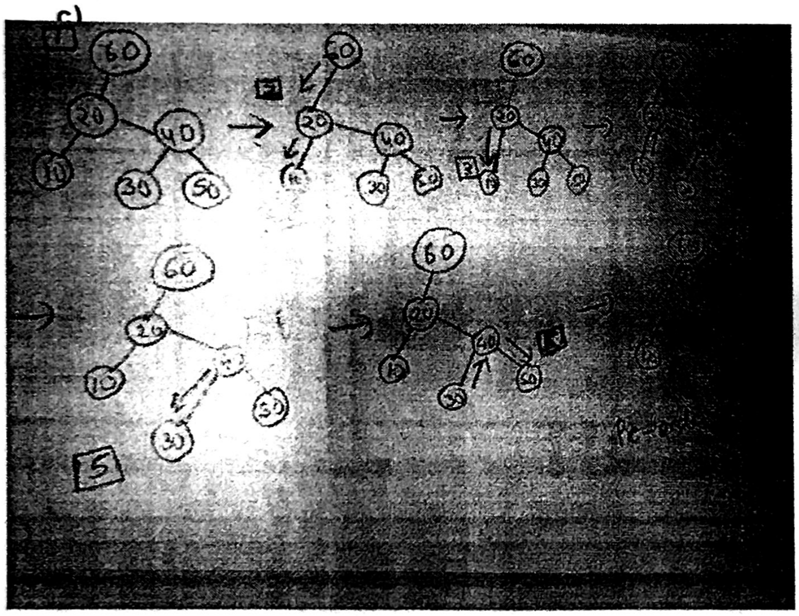
4.a)

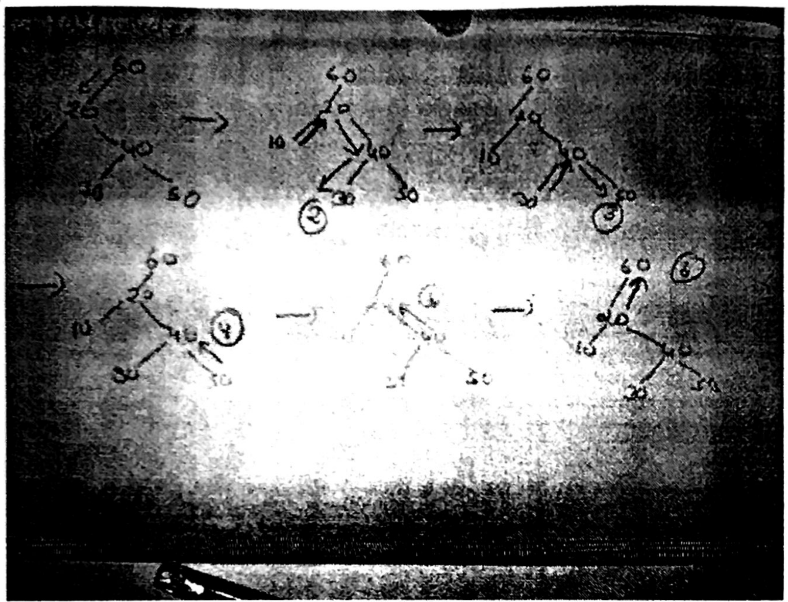
Binary Search Tree are a special form of binary tree where every node L<C<R (left Centre Right nodes). It is a ordered or sorted binary tree where the node on the left of any node will be smaller than it, while the node at the right of any node will be larger than it.

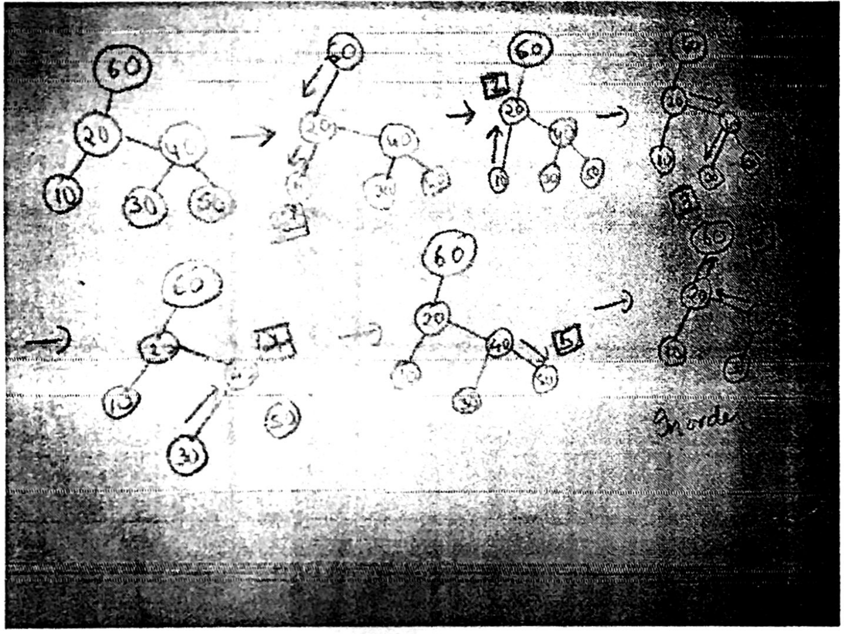
b)

PRE	60,20,10,40,30,50
IN	10,20,30,40,50,60
POST	10,30,50,40,20,60

c)







d)

```
int printOddValDesc(btnode *node){
     if(node==NULL)
          return;
/*Since it is a BST, and we need to print in descending order, go
towards right first because L<C<R */
     printOddValDesc(node->right);
     if(node->item%2==1)
          printf("%d,",node->item);
     printOddValDesc(node->left);
}
```