جامعة بنـها – كلية العلوم – قسم الرياضيات

مادة من المستوي الثاني (علوم حاسب- رياضيات)

يـوم الامتحان: الاحد

تـاريخ الامتحان: ١٨ / ١ / ٢٠١٥ م

المادة : هياكل بيانات (٢٥٢ رس)

الممتحن: د/ مصعب عبد الحميد محمد حسان

مدرس بقسم الرياضيات بكلية العلوم

الاسئلة و نموذج الإجابة

ورقة كاملة

Benha University
Faculty of Science
Dept. of Mathematics

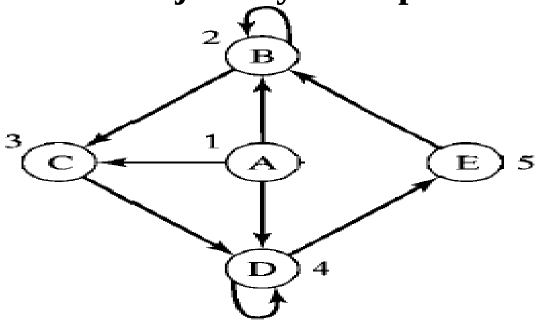
Time: Two Hours
First Semester 2014-2015
Date : 18/1/2015

Data Structures (MC252) for Second Level Students (Computer Science and Mathematics)

Answer the following questions

Question 1. (12 marks)

A- Given the following directed graph, write its adjacency-matrix representation and its adjacency-list representation. (4 marks)



B- Write a function to delete a node from a linked list. (4 marks)

C- Write the Dequeue function of the queue class. (4 marks)

Question 2. (12 marks)

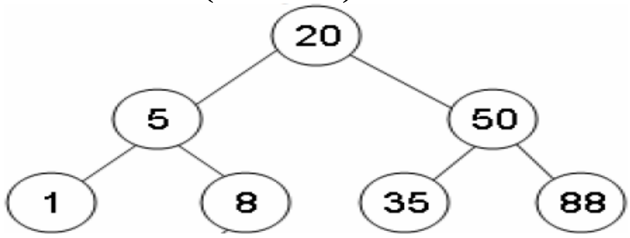
A- Draw the binary search tree by inserting the following sequence into an initially empty tree:

$< 25, \quad 11, \quad 31, \quad 4, \quad 15, \quad 45, \quad 26, \quad 14, \quad 16 >$

then write the post-order traversal of the tree. (4 marks)

B- Show the advantages and disadvantages of the linked list. (4 marks)

C- Draw the tree that results after inserting the value 29 and then removing the root node. (4 marks)



Question 3. (16 marks)

A- Write a function to concatenate two linked lists. (4 marks)

B- Write a function to insert a node in a given binary search tree. (6 marks)

C- Write the recursive search function for searching a binary search tree. (6 marks)

Question 4. (8 marks)

A- Write a function called findMin outside stack class that returns minimum item in a given stack using Pop and isEmpty functions. (4 marks)

B- Write a function to print the data items in a given binary search tree in increasing order. (4 marks)

Benha University
Faculty of Science
Dept. of Mathematics

Time: Two Hours
First Semester 2014-2015
Date : 18/1/2015

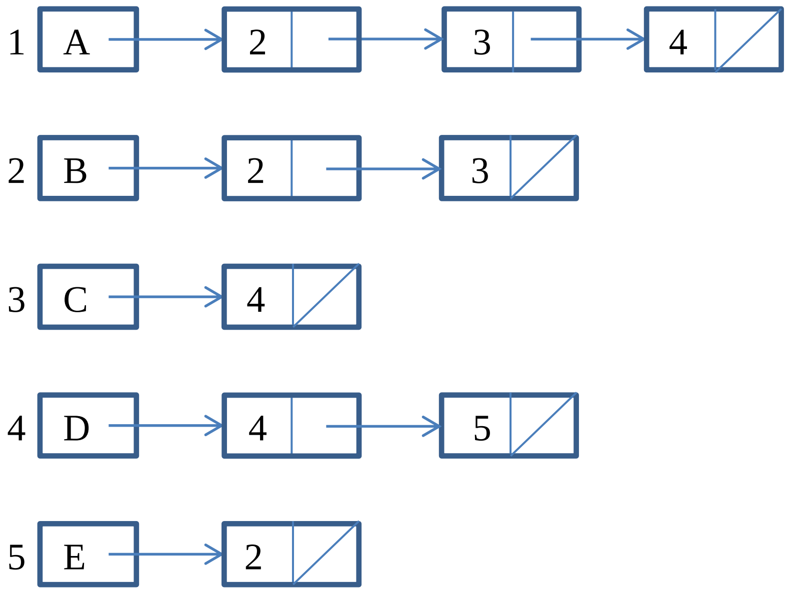
Data Structures (MC252) for Second Level Students (Computer Science and Mathematics)

Answer of Question 1.

A. Adjacency-Matrix Representation of the directed graph

0 1 1 1 0	1 A
0 1 1 0 0	2 B
0 0 0 1 0	3 C
0 0 0 1 1	4 D
0 1 0 0 0	5 E

Adjacency-List Representation of the directed graph



B- void Delete(int data1)

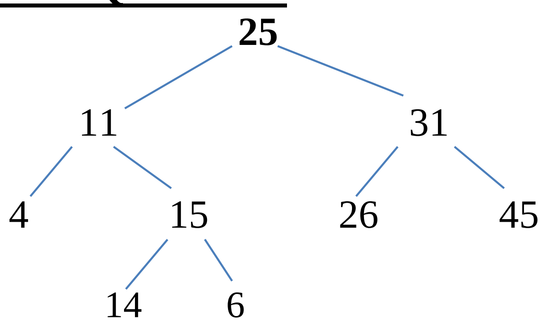
```
{

  node *temp;
  if (start->data == data1)
  {
    temp=start;
    start=start->next; /*First element deleted  (Case 1)*/
    delete temp;
    return;
  }


  node * q=start;
  while(q->next->next != NULL)
  {
```

```
    if(q->next->data == data1) /*Element deleted in between (Case 2)*/
    {
      temp=q->next;
      q->next=temp->next;
      delete temp;
      return;
    }
    q=q->next;
  }/*End of while */

  if(q->next->data == data1) /*Last element deleted (Case 3)*/
  {
    temp=q->next;
    delete temp;
    q->next=NULL;
    return;
  }

  cout << "\n\nElement " << data1 << " not found" << endl;

 }/*End of del()*/
```

C.
```
int Dequeue(){
      if(first ==  -1 || first > last)
          cout << "Queue is underflow " << endl;

      else{
          int item = queue_list[first];
          first = first + 1;
          return item;
      }
  }
```

Answer of Question 2.

A.



The post-order traversal of the tree is

4	14	16	15	11	26	4	5	31	25

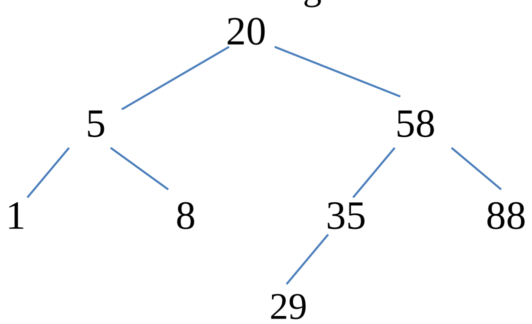
B. **Linked list have many advantages and some of them are:**

1. Linked list are dynamic data structure. That is, they can grow or shrink during the execution of a program.
2. Efficient memory utilization: In linked list (or dynamic) representation, memory is not pre-allocated. Memory is allocated whenever it is required. And it is removed when it is not needed.
3. Insertion and deletion are easier and efficient. Linked list provides flexibility in inserting a data item at a specified position and deletion of a data item from the given position.
4. Many complex applications can be easily carried out with linked list

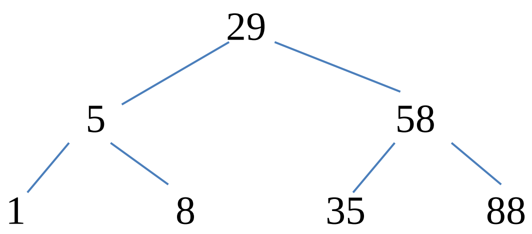
Linked list has following disadvantages

1. More memory: to store an integer number, a node with integer data and address field is allocated. That is more memory space is needed.
2. Access to an arbitrary data item is little bit cumbersome and also time consuming.

C. - The tree after inserting the value 29



The tree after removing the root node



<u>Answer of Question 3.</u>

A.

To concatenate two linked lists:

```
Concatenate(L, M):
  Node n = L -> start
  While (n -> Next != NULL) do
      n = n -> Next
   n -> Next  =  M -> start
   L1 = L
```

B. Function to insert a node in a given binary search tree.

```
void insert(int item)
{
        BinNode * parent = NULL;
        BinNode * locptr = myRoot;
        bool found = false;

        while(!found && locptr !=NULL)
        {
                parent = locptr;
                if(item < locptr->info)
                        locptr = locptr->left;
                else if(item > locptr->info)
                        locptr = locptr->right;
                else
                        found = true;

        }

        if(!found)
        {
                locptr = new BinNode();
                locptr->info = item;
                if(parent == NULL)  //empty tree
                    root = locptr;
                else if (item < parent->info)
                    parent->left = locptr;
                else
                    parent->right = locptr;
        }
        else
                cout<<"Item already in the tree \n";
}
```

C. The recursive search function for searching a binary search tree.

```
bool recursive(BinNode * subtreeRoot, int item)
{
    if (subtreeRoot == NULL)  // empty subtree
        return false;
    else{       //there is a nonempty subtree
        if (item < subtreeRoot->data)         // search left subtree
            return search_recursive(subtreeRoot->left, item);
        else if (item > subtreeRoot->data)    // search right subtree
            return search_recursive(subtreeRoot->right, item);
        else                                  // item is found
            return true;
    }

}
```

Answer of Question 4.

A.

```
int finMin(stack stack_obj){
        int min = 1000000;
        while(!stack_obj.isEmpty())
        {
          int y = stack_obj.Pop();
          if(y <= min)
            min = y;
        }
        return min;
    }
```

B. Print the data items in a given binary search tree in increasing order:

```
 void inorder(node * root_node)
 {
   if(root_node != NULL){
        inorder(root_node->data);
        cout << root_node->info << " " ;
        inorder(root_node->right);
   }
 }
```