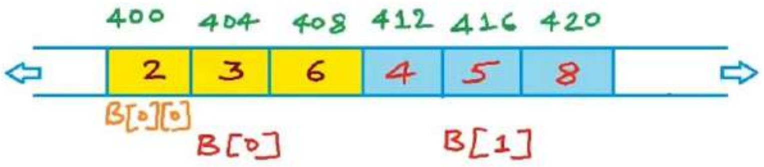
BBM 201
Exam 1
Time: 60 minutes

Answer the questions in the spaces provided on the question sheets.
KEEP YOUR CELLPHONE TURNED OFF UNTIL THE EXAM IS OVER.

Name: ______________________________

1. (15 points) If B is the array shown with its address above each node, write what the following lines of a program will print.

```
int B[2][3]
```



   (a) (3 points) `printf`("%d", &B[1][1]);

   (b) (3 points) `printf`("%d", B+1);

   (c) (3 points) `printf`("%d", *(*B+1));

   (d) (3 points) `printf`("%d", B[1]+2);

   (e) (3 points) `printf`("%d", *(B+1)+1);

2. (12 points) Provide the time complexities below for the worst-case.

   (a) (4 points) Write the time complexity of inserting and deleting an element of an array.

   (b) (4 points) Write the time complexity of inserting and deleting an element of a linked-list.

   (c) (4 points) Write the time complexity of push(x), pop(), top() and IsEmpty() operations for an element x of a stack.

3. (9 points) How many bytes do the following data structures occupy in the memory?

   (a) (3 points) An INTEGER array of size 5.

   (b) (3 points) A CHARACTER doubly linked-list of size 5.

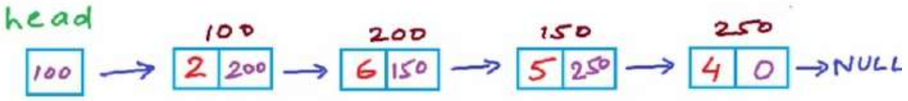
   (c) (3 points) A DOUBLE linked-list of size 5.

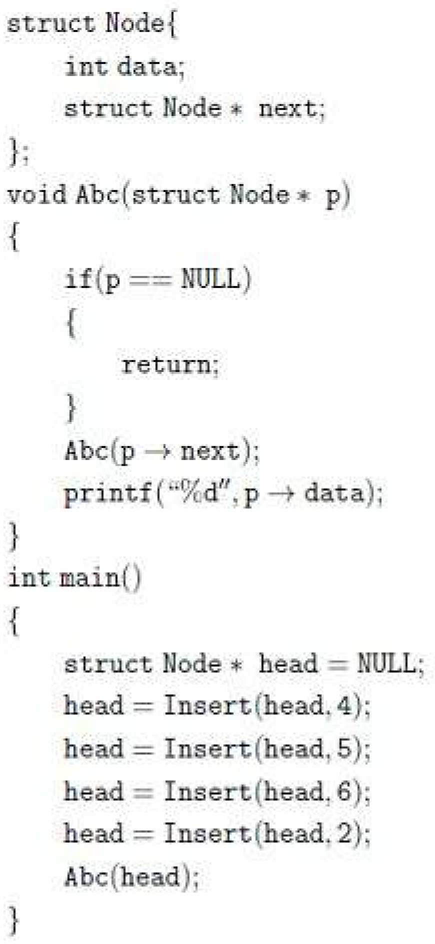
4. (12 points) (a) (6 points) Show how a stack looks after each of the following operations is applied consecutively. Write your answer in the stack column of the table below. Assume that this stack is initially empty.

   (b) (6 points) Write what `IsEmpty( )` and `Top( )` would return for the current stack if it was executed after each operation in the table below.

operation	current stack	IsEmpty( )	Top ( )
Push(2);			
Push(4);			
Pop( );			
Push(7);			
Push(3);			
Pop( );			

5. (16 points) (a) (12 points) Write each of the recursive calls made in their order when `Abc(head)` is executed in `main ()`. The address of each node is written above that node in this linked list.



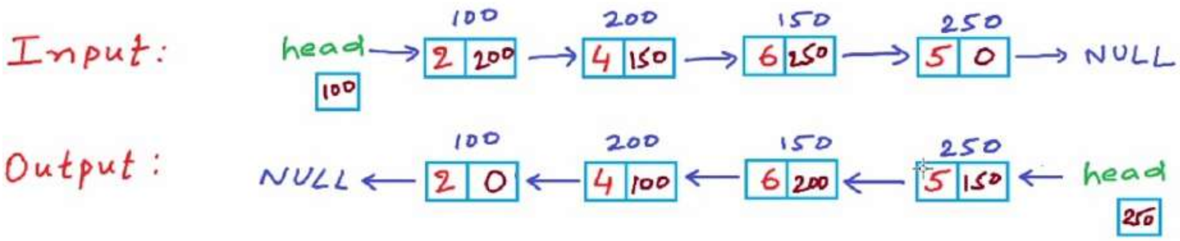


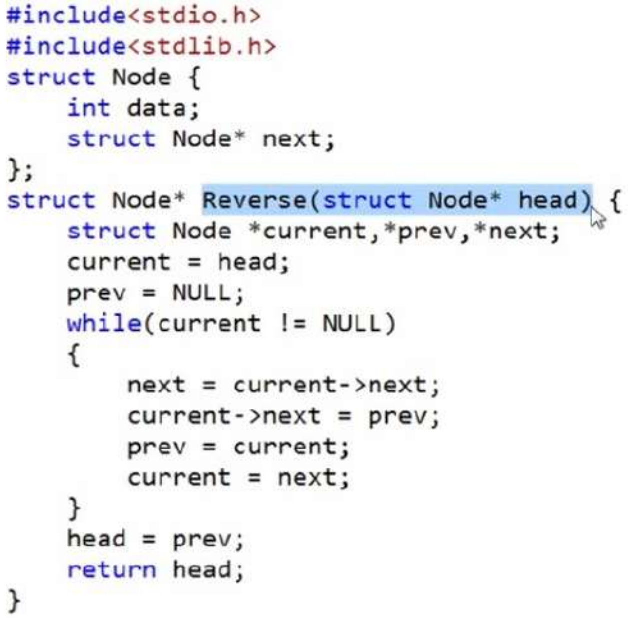
```
struct Node{
    int data;
    struct Node ∗ next;
};
void Abc(struct Node ∗ p)
{
    if(p == NULL)
    {
        return;
    }
    Abc(p → next);
    printf(“%d”, p → data);
}
int main()
{
    struct Node ∗ head = NULL;
    head = Insert(head, 4);
    head = Insert(head, 5);
    head = Insert(head, 6);
    head = Insert(head, 2);
    Abc(head);
}
```

(b) (4 points) Write what `main()` will return in this program program.
(Note: `Insert(head, data)` inserts a new node in the beginning by putting `data` as the value of the head node.)

6. (36 points) The Reverse function below reverses any given linked list by using iteration method. The input argument of the Reverse function is the head of the linked list. In the following linked list, the number above each node indicates its address in the memory.

   (a) (20 points) If we execute the Reverse function for this linked list, redraw the changes on the input linked list after EACH ITERATION STEP of the while loop by showing the changes on the NODES and on the LINKS (arrows).





```
#include<stdio.h>
#include<stdlib.h>
struct Node {
    int data;
    struct Node* next;
};
struct Node* Reverse(struct Node* head) {
    struct Node *current,*prev,*next;
    current = head;
    prev = NULL;
    while(current != NULL)
    {
        next = current->next;
        current->next = prev;
        prev = current;
        current = next;
    }
    head = prev;
    return head;
}
```

   (b) (16 points) Write the value of addresses stored in "current", "prev" and "next" initially and after each iteration step of the while loop in the table below.

iteration	current	prev	next
1			
2			
3			
4			
5			

BBM 201

Exam 1

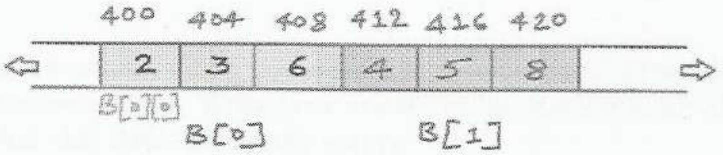
Time: 60 minutes

> Answer the questions in the spaces provided on the question sheets.
> KEEP YOUR CELLPHONE TURNED OFF UNTIL THE EXAM IS OVER.

Name: SOLUTIONS

1. (15 points) If B is the array shown with its address above each node, write what the following lines of a program will print.

   `int B[2][3]`

   (a) (3 points) `printf("%d", &B[1][1]);` 416

   (b) (3 points) `printf("%d", B+1);` 412

   (c) (3 points) `printf("%d", *(*B+1));` 3

   (d) (3 points) `printf("%d", B[1]+2);` 420

   (e) (3 points) `printf("%d", *(B+1)+1);` 416

2. (12 points) Provide the time complexities below for the worst-case.

   (a) (4 points) Write the time complexity of inserting and deleting an element of an array.

   Insert: O(n) Delete: O(n)

   (b) (4 points) Write the time complexity of inserting and deleting an element of a linked-list.

   Insert: O(n) Delete: O(n)

   (c) (4 points) Write the time complexity of push(x), pop(), top() and IsEmpty() operations for an element x of a stack.

   All are O(1)

3. (9 points) How many bytes do the following data structures occupy in the memory?

   (a) (3 points) An INTEGER array of size 5.

   (b) (3 points) A CHARACTER doubly linked-list of size 5.

   (c) (3 points) A DOUBLE linked-list of size 5.

a) $4 \times 5 = 20$

b) $(1 + 2 \times 4) * 5 = 45$

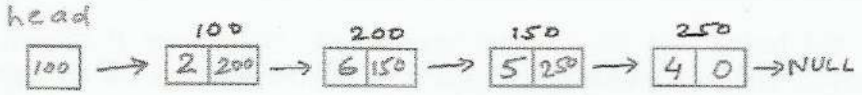
c) $(8 + 4) \cdot 5 = 60$

4. (12 points) (a) (6 points) Show how a stack looks after each of the following operations is applied consecutively. Write your answer in the stack column of the table below. Assume that this stack is initially empty.

   (b) (6 points) Write what `IsEmpty( )` and `Top( )` would return for the current stack if it was executed after each operation in the table below.

operation	current stack	IsEmpty( )	Top ( )
Push(2);	2	False	2
Push(4);	2, 4	"	4
Pop( );	2	"	2
Push(7);	2, 7	"	7
Push(3);	2, 7, 3	"	3
Pop( );	2, 7	"	7

5. (16 points) (a) (12 points) Write each of the recursive calls made in their order when Abc(head) is executed in main (). The address of each node is written above that node.



```
struct Node{
    int data;
    struct Node * next;
};
void Abc(struct Node * p)
{
    if(p == NULL)
    {
        return;
    }
    Abc(p → next);
    printf("%d", p → data);
}
int main()
{
    struct Node * head = NULL;
    head = Insert(head, 4);
    head = Insert(head, 5);
    head = Insert(head, 6);
    head = Insert(head, 2);
    Abc(head);
}
```

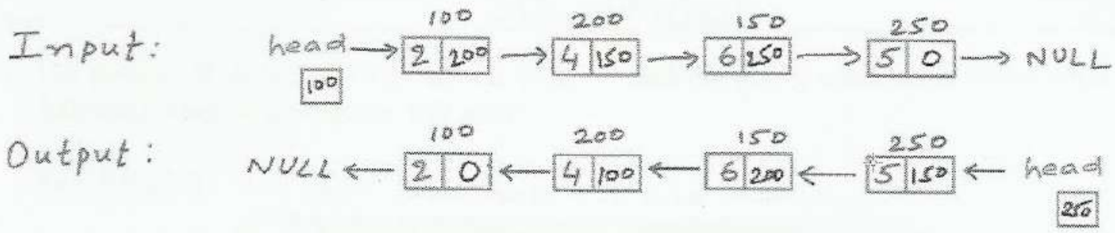
ABC(100)
↓
ABC(200)
↓
ABC(150)
↓
ABC(250)
↓
ABC(NULL)

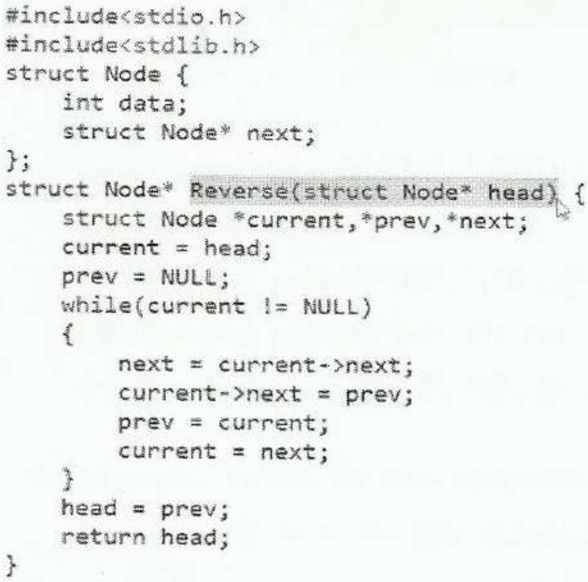
(b) (4 points) Write what main() will return in this program program.
(Note: Insert(head, data) inserts a new node in the beginning by putting data as the value of the head node.)

4 5 6 2

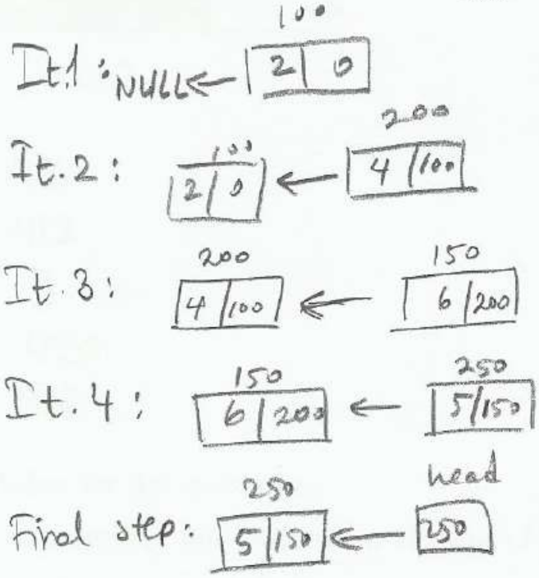
6. (36 points) The Reverse function below reverses any given linked list by using iteration method. The input argument of the Reverse function is the head of the linked list. In the following linked list, the number above each node indicates its address in the memory.

(a) (20 points) If we execute the Reverse function for this linked list, redraw the changes on the input linked list after EACH ITERATION STEP of the while loop by showing the changes on the NODES and on the LINKS (arrows).





```
#include<stdio.h>
#include<stdlib.h>
struct Node {
    int data;
    struct Node* next;
};
struct Node* Reverse(struct Node* head) {
    struct Node *current,*prev,*next;
    current = head;
    prev = NULL;
    while(current != NULL)
    {
        next = current->next;
        current->next = prev;
        prev = current;
        current = next;
    }
    head = prev;
    return head;
}
```



(b) (16 points) Write the value of addresses stored in "current", "prev" and "next" initially and after each iteration step of the while loop in the table below.

iteration	current	prev	next
1	100	NULL	×
2	200	100	200
3	150	200	150
4	250	150	250
5	NULL	250	NULL

BBM 201

Final Exam

Time: 60 minutes

Answer the questions in the spaces provided on the question sheets.
KEEP YOUR CELLPHONE TURNED OFF UNTIL THE EXAM IS OVER.

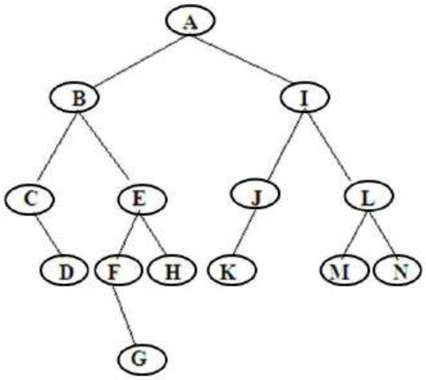
Name: ____

1. For the tree below, write

   (a) (6 points) the preorder traversal,

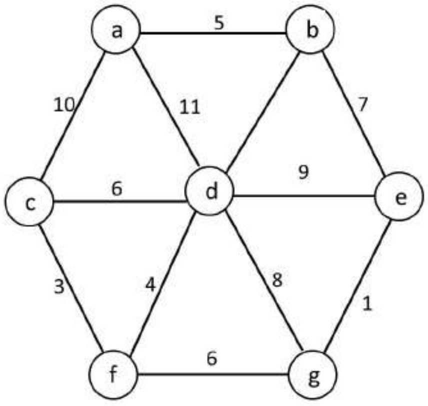
   (b) (6 points) the inorder traversal,

   (c) (6 points) the postorder traversal.

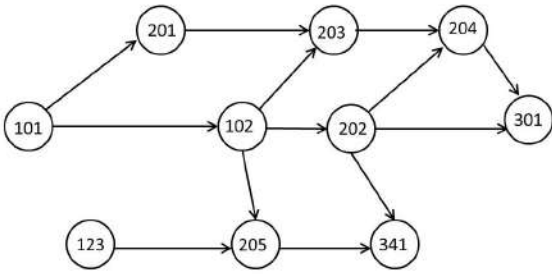


2. (12 points) Show the output of Prim's algorithm for this graph using start vertex a. Show your work by filling the table for each added vertex.

step	edge added	Weights compared
1		
2		
3		
4		
5		
6		



3. (a) (10 points) Find a topological sort of the graph below using depth first search. **Explain how depth first search is used to give your answer.**



(b) (10 points) Find a topological sort of this graph above using an implementation that uses enqueue and dequeue of vertices. **Show the queue at each step of this algorithm.**

4. The input argument of the function “Abc” is the root of a tree, whose vertices are defined using “Node” with fields “left”, a pointer to the left child, and “right”, a pointer to the right child.

```
 Line1   struct Node{
 Line2       char data;
 Line3       struct Node * left;
 Line4       struct Node * right;
 Line5   };
 Line6   void Abc(struct Node * root)
 Line7   {
 Line8       if(root == NULL)return;
 Line9       printf(“%c″, root → data);
Line10       Abc(root → left);
Line11       Abc(root → right);
Line12   }
```

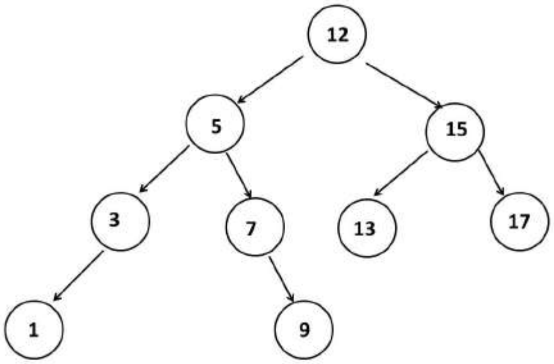
Which lines should be interchanged so that the function Abc gives the following traversals? (It may be also a valid answer not to interchange any lines.)

(a) (3 points) preorder traversal

(b) (3 points) inorder traversal

(c) (3 points) postorder traversal

5. (12 points) Remove the values 15, 5 and 3 from this tree <u>in this order</u>. **Show the remaining tree after each removal**.

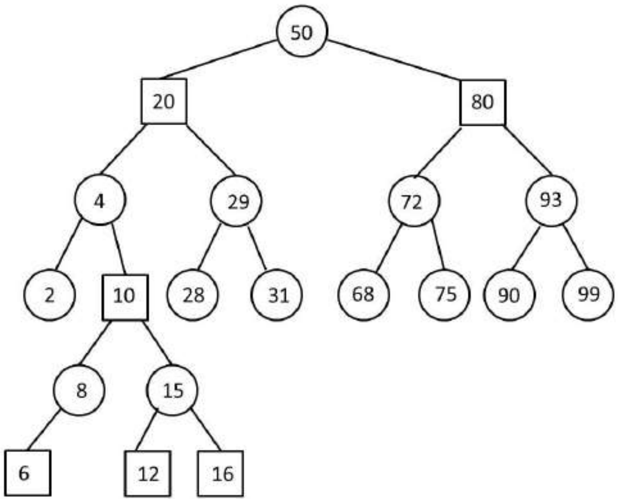


6. Consider the balanced red and black tree, where nodes in □ are red and nodes in ○ are black. Apply each operation to the tree below and show each step by showing the changes on the position and the color of the nodes.

   (a) (3 points) Add 5

   (b) (4 points) Add 7

   (c) (6 points) Delete 2



7. Write the **worst-case** running time of the following implementations using big Oh notation.

   (a) (2 points) Searching a given value on a binary search tree on n nodes with height $n/2$

   (b) (2 points) Finding the minimum value on a binary search tree on n nodes with height $\sqrt{n}$

   (c) (2 points) Adding a new value to a balanced binary search tree on n nodes

   (d) Traversal of all vertices in a graph G with V vertices and E edges using breadth first search when

      (a) (2 points) using adjacency list

      (b) (2 points) using adjacency matrix

   (c) (2 points) Finding minimum-cost paths from a single source vertex using Dijkstra's algoritm in a graph G with V vertices and E edges.

   (d) Finding a topological sort for a DAG, G, with V vertices and E edges with an implementation that

      (a) (2 points) uses a queue

      (b) (2 points) uses depth first search

BBM 201

Final Exam

Time: 60 minutes

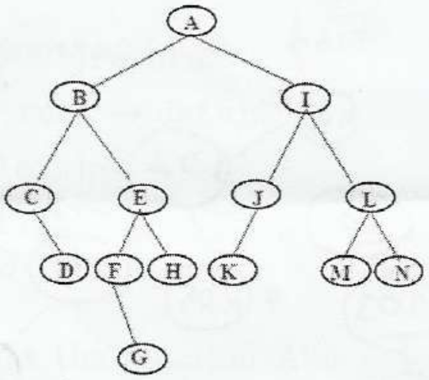
Answer the questions in the spaces provided on the question sheets.
KEEP YOUR CELLPHONE TURNED OFF UNTIL THE EXAM IS OVER.

Name: SOLUTIONS

1. For the tree below, write

   (a) (6 points) the preorder traversal,

   ABCDEFGHIJKLMN



   (b) (6 points) the inorder traversal,

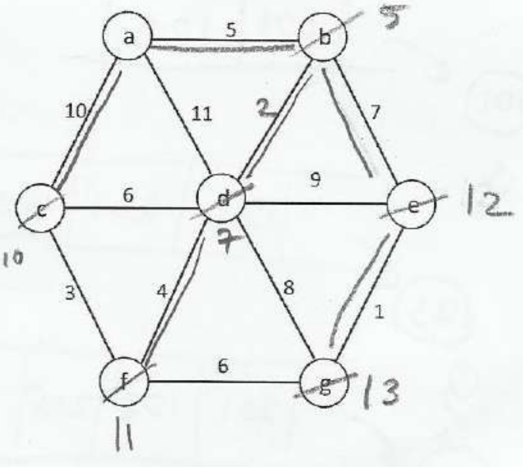
   CDBFGEHAKJIMLN

   (c) (6 points) the postorder traversal.

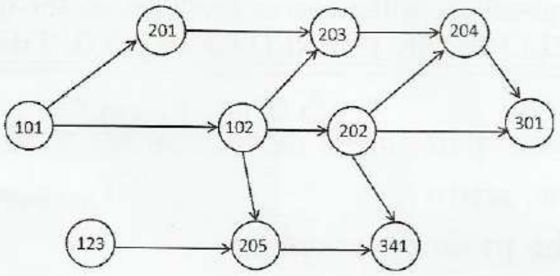
   DCGFHEBKJMNLIA

2. (12 points) Show the output of Prim's algorithm for this graph using start vertex a. Show your work by filling the table for each added vertex.

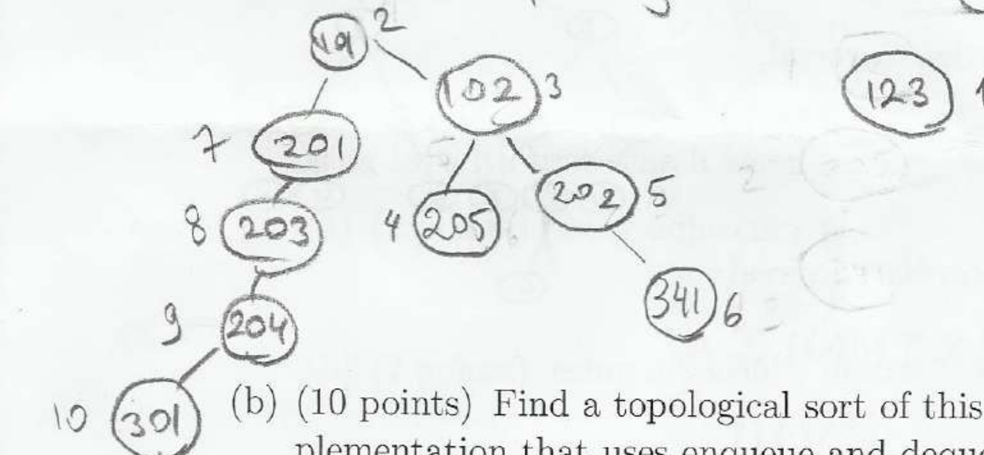
step	edge added	Weights compared
1	ab	5, 10, 11
2	bd	7, 10, 11, 12
3	ac	10, 13, 12, 16, 11, 15
4	df	12, 13, 15, 16
5	be	15, 17, 16
6	eg	15, 17



3. (a) (10 points) Find a topological sort of the graph below using depth first search. **Explain how depth first search is used to give your answer.**



Find a spanning Forest using DFS:



123, 101, 102, 205, 202, 341, 201, 203, 204, 301

(b) (10 points) Find a topological sort of this graph above using an implementation that uses enqueue and dequeue of vertices. **Show the queue at each step of this algorithm.**

Q

101	123

101

Q

123	201	102

123

Q

201	102	205

201

Q

102	205	203

102

Q

205	203	202

205

Q

203	202	341

203

Q

202	341	204

202

Q

341	204	301

341, 204, 301

Sort: 101, 123, 201, 102, 205, 203, 202, 341, 204, 301

4. The input argument of the function “Abc” is the root of a tree, whose vertices are defined using “Node” with fields “left”, a pointer to the left child, and “right”, a pointer to the right child.

```
 Line1  struct Node{
 Line2      char data;
 Line3      struct Node * left;
 Line4      struct Node * right;
 Line5  };
 Line6  void Abc(struct Node * root)
 Line7  {
 Line8      if(root == NULL)return;
 Line9      printf(“%c″, root → data);
Line10      Abc(root → left);
Line11      Abc(root → right);
Line12  }
```

Which lines should be interchanged so that the function Abc gives the following traversals? (It may be also a valid answer not to interchange any lines.)

(a) (3 points) preorder traversal

No change

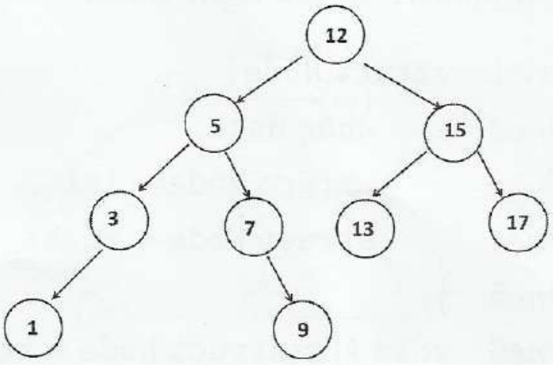
(b) (3 points) inorder traversal

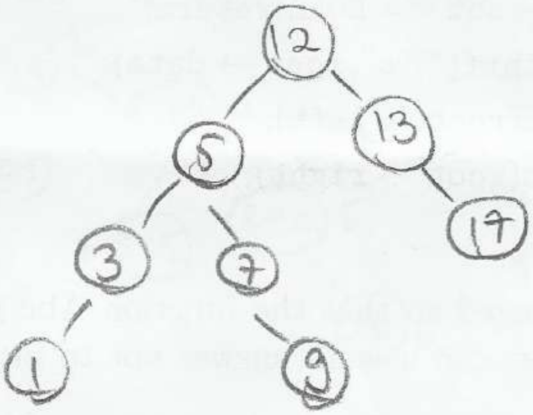
$$\left.\begin{array}{l}\text{Line } 9\\ 10\\ 11\end{array}\right\} \longrightarrow \begin{array}{l}\text{Line } 10\\ 9\\ 11\end{array}$$

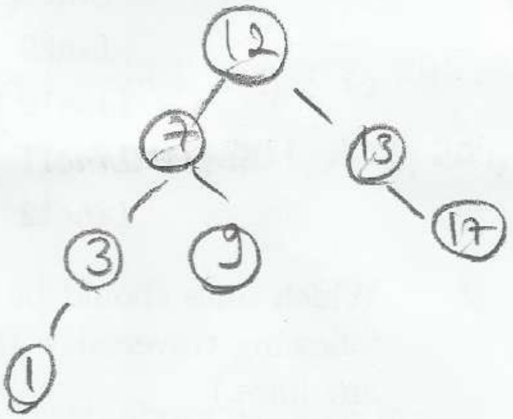
(c) (3 points) postorder traversal

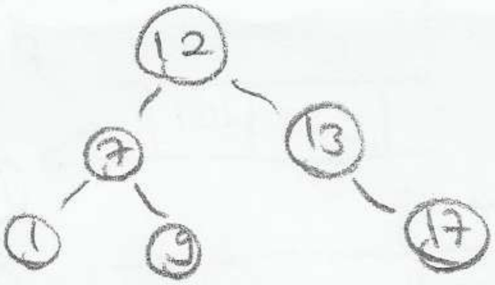
$$\left.\begin{array}{l}\text{Line } 9\\ 10\\ 11\end{array}\right\} \longrightarrow \begin{array}{l}\text{Line } 10\\ 11\\ 9\end{array}$$

5. (12 points) Remove the values 15, 5 and 3 from this tree in this order. **Show the remaining tree after each removal.**







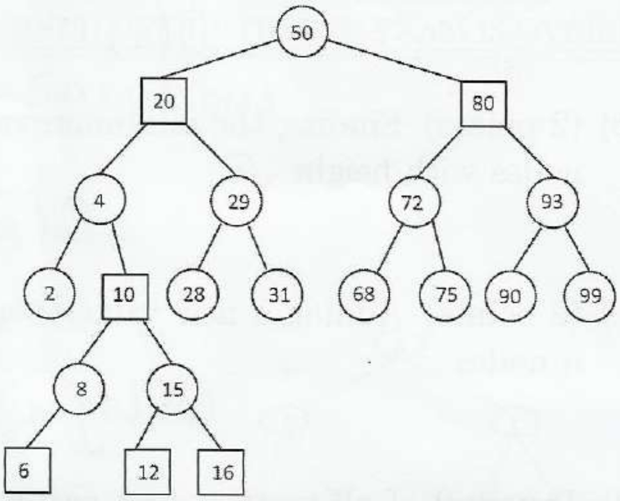


6. Consider the balanced red and black tree, where nodes in □ are red and nodes in ○ are black. Apply each operation to the tree below and show each step by showing the changes on the position and the color of the nodes.

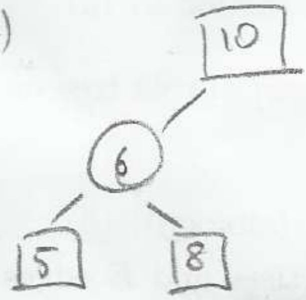
(a) (3 points) Add 5

(b) (4 points) Add 7

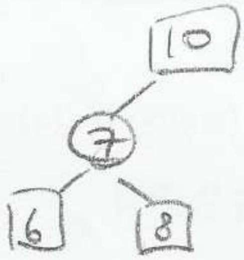
(c) (6 points) Delete 2



a)

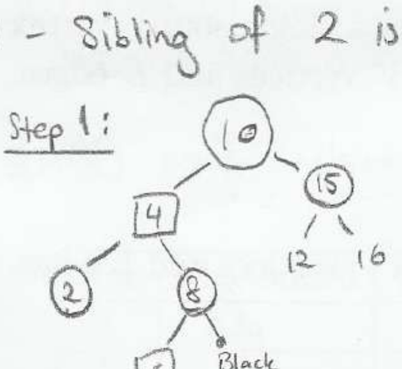


b)



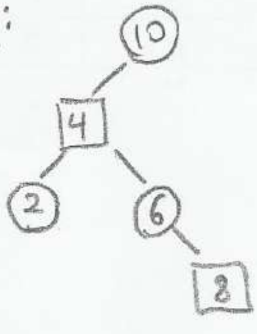
c) - Sibling of 2 is not black

Step 1:

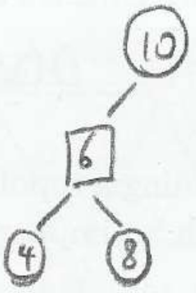


- Further away (from 2) child of 8 is black

Step 2:



- Rotation toward 2:



7. Write the **worst-case** running time of the following implementations using big Oh notation.

   (a) (2 points) Searching a given value on a binary search tree on n nodes with height $n/2$

   $O(n)$

   (b) (2 points) Finding the minimum value on a binary search tree on n nodes with height $\sqrt{n}$

   $O(\sqrt{n})$

   (c) (2 points) Adding a new value to a balanced binary search tree on n nodes

   $O(\log n)$

   (d) Traversal of all vertices in a graph G with V vertices and E edges using breadth first search when

      (a) (2 points) using adjacency list

      $O(V+E)$

      (b) (2 points) using adjacency matrix

      $O(V^2)$

   (c) (2 points) Finding minimum-cost paths from a single source vertex using Dijkstra's algoritm in a graph G with V vertices and E edges.

   $O(V^2+E) = O(V^2)$ (better implem. gives $O((V+E)\log V)$

   (d) Finding a topological sort for a DAG, G, with V vertices and E edges with an implementation that

      (a) (2 points) uses a queue

      $O(V+E)$

      (b) (2 points) uses depth first search

      $O(V+E)$