BBM201 – Data Structures – Fall 2016
Single Course Exam
7.10.2016

Name Surname: _____

Student ID : _____ **Section:**_____

Duration: …. minutes

Question	1	2	3	4	5	6	7	8	9	10	*	Total
Points	10	9	10	7	12	8	12	12	10	10	10	100
Grade												

Question 1. Decide if the following statements are true or false, circle your answer. Give a short explanation if you chose 'False'.

a) The worst-case cost to insert an element to the beginning of a linked list of size n is O(1).

True ☐ False ☐

b) The worst-case cost to insert an element to the end of an array of size n is O(n), if there is still space at the end of the array.

True ☐ False ☐

c) The worst-case cost to delete an element from a linked list of size n is O(1).

True ☐ False ☐

d) The worst-case cost to delete an element from an array of size n is O(n).

True ☐ False ☐

e) The memory used to store a linked list of 5 integers is 40 bytes (suppose each pointer is 4 bytes).

True ☐ False ☐

Question 2. Mark the following statements as True or False. Provide a short explanation for your answer. Each wrong answer cancels out a correct one.

(a) $3\ n^2 + 10\ n \log n = \Omega(n^2)$

(b) $3\ n^2 + 10\ n \log n = O(n \log n)$

(c) $n^{100} + 2^n = O(n^{100})$

(d) $n = o(n^2)$

Question 3. Let A[n][n] be a lower triangular matrix (see the example given below).

The elements of this triangular matrix are stored in a one-dimensional array as given below:

a_{oo}	0	0	0
a_{10}	a_{11}	0	0
a_{20}	a_{21}	a_{22}	0
a_{30}	a_{31}	a_{32}	a_{33}

U

a_{00}	a_{10}	a_{11}	a_{20}	a_{21}	a_{22}	a_{30}	a_{31}	a_{32}	a_{33}

a) What is the number of items stored in U if A has n rows and n columns?

b) What is x if U[x] stores the entry A[i][j]? Show your work.

c) Please fill in the blanks in the method *readtriangularmatrix(int U[], int n) that reads integers from the keyboard and fills the one-dimensional array U with entries of A as shown in the example.*

```
void readtriangularmatrix(int U[], int n){

    int i, j, k;
    if(n*(n+1)/2 > MAX_SIZE){
       printf("\n invalid array size \n");
       exit(-1);
    }
    else
      for(i=0; i<=n-1; i++){
          k=……..
          for(j=0; j<=i; j++)
                scanf("%d", ……);
       }
}
```

Question 4. Let 48+2*35-+47-* be an expression given in postfix notation (suppose each number has only one digit again). Fill the table on the right accordingly, if the

input is stored and processed using a stack notation in order to evaluate the given expression. ([0],[1],[2] indicate the position number of an element, [0] being the bottom position.)

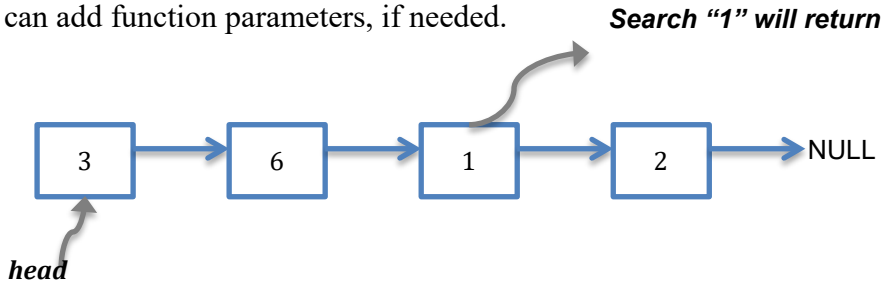
Token	Stack			Top
	[0]	[1]	[2]	
4				
8				
+				
2				
*				
3				
5				
-				
+				
4				
7				
-				
*				

Question 5. Write a recursive method to free a given linked list.

```
struct node{
      int data;
      struct node* next;
};

void freeList(struct node* head){









}
```

Question 6.

Write a **recursive** method that searches for a value in a linked list and returns the position of the value in the linked list if found, otherwise returns -1. The first item in the linked list has position 1. You can add more parameters for the given method, if needed. (Using global variable is **not** allowed!)
You can add function parameters, if needed.



```
struct Node{
      int data;
      struct Node* next;
};


int  SearchItem(struct Node* head, int value              )
{
```

BBM201 – Data Structures – Fall 2017 - Final Exam
10.01.2018 – 120 minutes

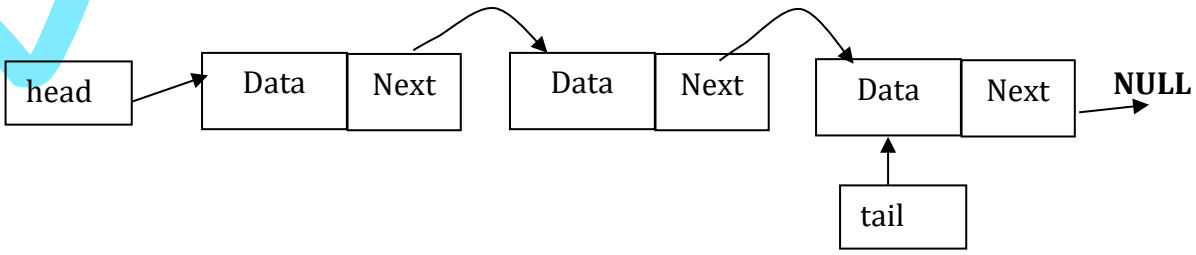
Name Surname: _____

Student ID : _____

Section: ☐ **Section 1 (Burcu Can)**
☐ **Section 2 (Sevil Şen)**
☐ **Section 3 (Adnan Özsoy)**

Questions	1	2	3	4	5	6	7	Total
Points	16	15	10	12	12	20	15	100
Grade								

1. (Linked List) ***(16 points)*** 1. Consider the linked list shown below. Please write the function named ***removeTail*** that will remove the node named by "tail". Please handle special cases in your method.




```
typedef struct node{
          int Data;
      struct node *Next;
} nodeLL;

nodeLL *head, *tail;

void removeTail()
{
```

```
}
```

2. (Arrays) ***(15 points)***

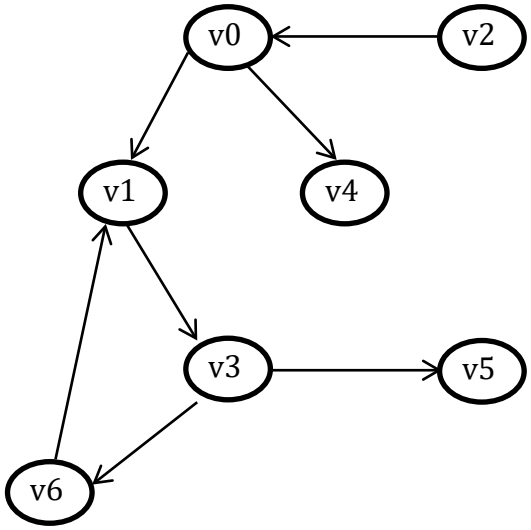
```
#include <stdio.h>
#define ROW 3
#define COL 4

void change(int array[][COL])
{
    int i, l, total;
    for(i = 0; i < ROW; i++){
        total = 0;
        for(l = 0; l < COL; l++){
            if(i == l)
                continue;
            total += array[i][l];
        }
        array[i][i] = total;
    }
}

   int main(void)
{
    int array[ROW][COL] = {{1,2,3,4}, {5,6,7,8}, {9,10,11,12}};
    int i, l;
    change(array);
    for(i = 0; i < ROW; i++){
        for(l = 0; l < COL; l++)
            printf("%d ", array[i][l]);
        printf("\n", array[i][l]);
    }
             return 0;
}
```

a. What does the ***change*** function do? Please explain.

b. Please re-write the same function **by using only point arithmetic**? The new function called ***change2*** is given below. Please fill out the blank boxes in the code. Let's assume that this function is called as ***change2(&dizi[0][0]).***

```
void change2(_____________)
{
    int *temp  = _________;
    int i, l, total;
    for(i = 0; i < ROW; i++){
        total = 0;
        for(l = 0; l < COL; l++, _______){
            if(i == l)
                continue;
            total += _________;
        }
        _______________ = total;
    }
}
```

3. (Graphs) ***(10 points)*** Consider this directed graph:

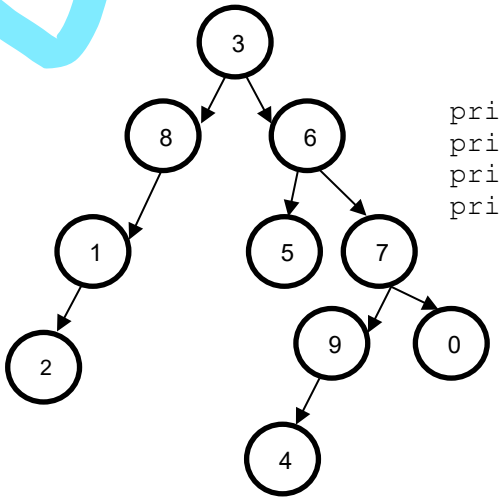


a. In what order are the vertices visited for a depth-first search that starts at v0?

b. In what order are the vertices visited for a breadth-first search that starts at v0?

Note : If a node v has more than one adjacent nodes pointing from v, please assume that they are going to be processed in increasing order according to their numbers. For example, for the adjacent nodes pointing from v3, firstly v5 is going to be processed, then v6.

4. (Trees) ***(12 points)*** Write a recursive method that prints the level of a given node in a binary tree. If the node is not found, it does not print anything. Assume all numbers are unique in the tree. Iterative solutions will not be accepted.




```
printLevel(root, 8, 0); // It
prints "2"
printLevel(root, 4, 0); // It prints "5"
printLevel(root, 9, 0); // It prints "4"
printLevel(root, 12, 0);//It prints nothing
```

```
struct node
{
  int value;
  struct node *left;
  struct node *right;
};


void printLevel(struct node* root, int number, int level){









}
```

5. (Recursion and Performance Analysis) ***(12 points)*** According to the below recursive method, answer the following questions.

```
int recurse(int n)
{
    if (n <= 0)
        return 1;
    else
        return 1 + recurse(n/2);
}
```

a) What does the method return for n=8?

b) How many times will your method be called recursively for n=256?

c) What is the algorithm complexity of the recursive method in big Oh notation?

d) What is the total variable space needed for the execution of the method for an input size of 256, if the integer size is 4 bytes?

6. (Stack & Queues) ***(20 points)*** Complete the linked list stack implementation below for spaces "_____" as needed and inside the functions, and give the output of the code at the end.

```
typedef struct Node{
  struct _________ next;
  int data;
} Node;

typedef struct stackT{
  _____________ top;
  int maxSize; //max number of items allowed in the stack
  int count;  // current number of items in the stack
} stackT;

int StackIsEmpty(__________mystack)
{



}
int StackIsFull(__________mystack)
{




}
void StackPush( _________ mystack, int element)
{
  if (StackIsFull(________)) {
    printf("Can't push element on stack: stack is full.\n");
    exit(1);  /* Exit, returning error code. */
  }
  /* FILL BELOW, Put information in stack; update top and count. */









}
int StackPop(___________mystack)
{
  if (StackIsEmpty(________)) {
    printf(" Can't pop element stack is empty.\n");
    exit(1);  /* Exit, returning error code. */
  }
  /* update top and count, remove node from stack; return data of the node */









  return _________;
}
```

CONTINUE NEXT PAGE→

```
int main()
{
        stackT mystack;     /* A stack to hold ints. */
        mystack.maxSize=10;
        mystack.count=0;
        mystack.top=NULL;

        StackPush(____mystack, 5);
        StackPush(____mystack, 1);
        StackPush(____mystack, 7);

        printf(StackPop(____mystack));
        printf(StackPop(____mystack));
        printf(StackPop(____mystack));
        printf(StackPop(____mystack));

        return 1;
}
```

//Output of the above code : _____

7. a-(Tries) ***(8 points)*** Construct a trie from given key-value pairs:

cell	1
cope	5
call	7
caller	9
seller	8
sell	2
set	4
cop	3

b- (**Data Structures**) ***(7 points)*** Which data structure is the most appropriate for the following problems? No explanation is needed. (Possible answers: stack, queue, array, multidimensional array, linked list, circular linked list, sparse matrix, tree, trie, graph, hash table),

(a) When a printer receives several printing requests, _____

(b) Map of highway system used to display traffic travel times on a web page. The map displays principle cities, intersections, and major landmarks, the roads that connect them, and the travel times between them along those roads.

_____

(c) Chess board – an 8 x 8 board used for a game of chess. Each square on the board is either empty or contains a chess piece. _____

(d) When you want to store a collection of items in a set where the size is known.____

_____

(e) A list of the legal words in a Scrabble game. We want to be able to quickly check whether words used by players do, in fact, exist in the list. _____

(f) When you want to build a scheduler for processes where each process takes CPU cycles in rounds again and again, _____

(g) When it is checked by the computer if a list of parentheses is balanced,

_____

1. Given an array, rotate the array to the right by *k* steps, where *k* is non-negative.

   Example:

   Input: `[1,2,3,4,5,6,7] and k = 3, numsSize=7`
   Output: `[5,6,7,1,2,3,4]`

   Please try not to use any extra array.

```
void rotate(int* nums, int numsSize, int k) {









    }
```

2. Write a function to delete a node (except the tail) in a singly linked list, given only access to that node.

Example:

Input: head = [4,5,1,9], node = 5
Output: [4,1,9]

```
void deleteNode(struct ListNode* node) {




}
```

BBM 201
Final Exam
Time: 60 minutes

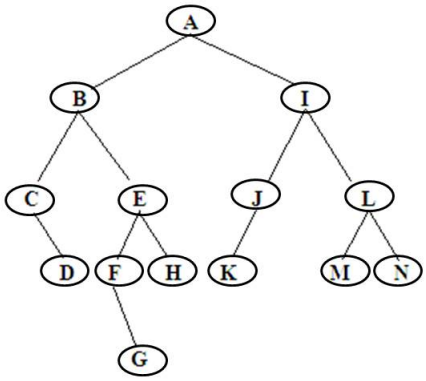
Answer the questions in the spaces provided on the question sheets. KEEP YOUR CELLPHONE TURNED OFF UNTIL THE EXAM IS OVER.

Name: _____

1. For the tree below, write
   (a) (6 points) the preorder traversal,

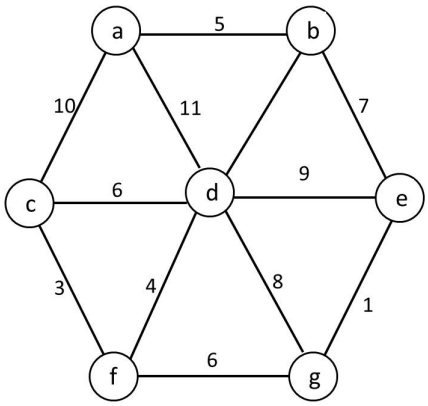
   (b) (6 points) the inorder traversal,

   (c) (6 points) the postorder traversal.

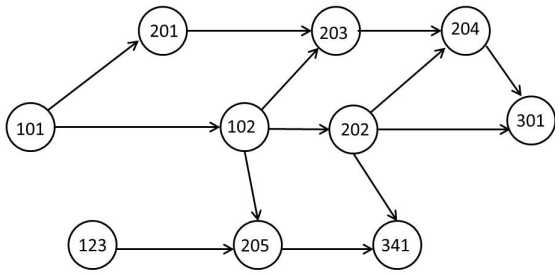


2. (12 points) Show the output of Prim's algorithm for this graph using start vertex a. Show your work by filling the table for each added vertex.

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  }
```

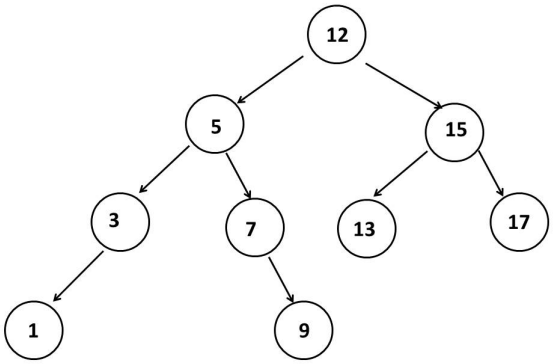
Which lines should be interchanged so that the function Abc gives the following traversals? (It may be also a valid answer not to interchange any lines.)

(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**.

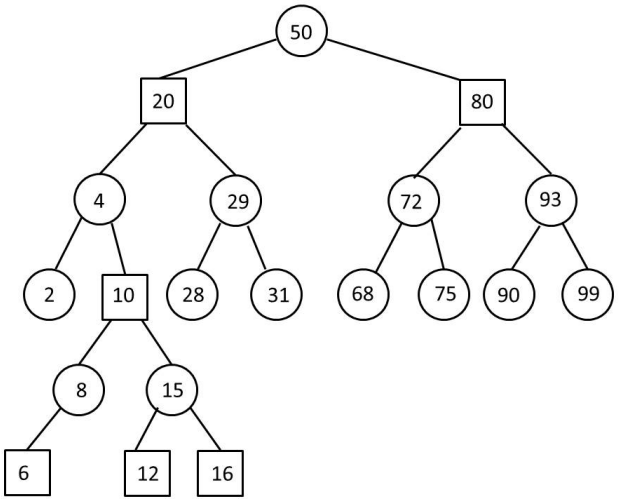


6. Consider the balanced red and black tree, where nodes in □ are red and nodes in ◯ are black. Apply each operation to the tree below and show each step by showing the changes on the position and the color of the nodes.

   (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

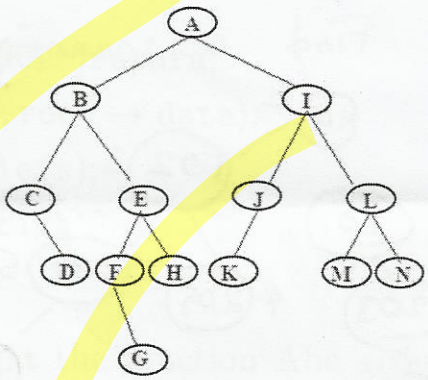
Answer the questions in the spaces provided on the question sheets.
KEEP YOUR CELLPHONE TURNED OFF UNTIL THE EXAM IS OVER.

Name: SOLUTIONS

1. For the tree below, write

   (a) (6 points) the preorder traversal,

   ABCDEFGHIJKLMN



   (b) (6 points) the inorder traversal,

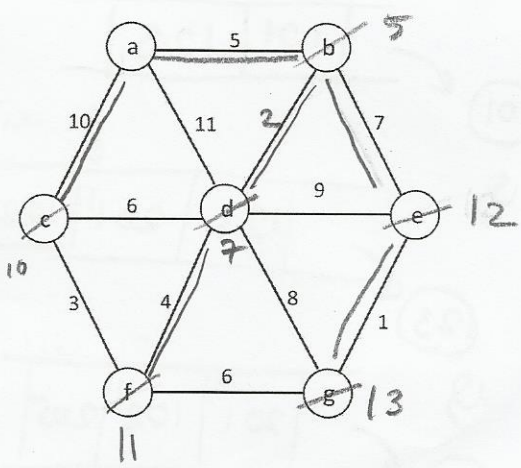
   CDBFGEHAKJIMLN

   (c) (6 points) the postorder traversal.

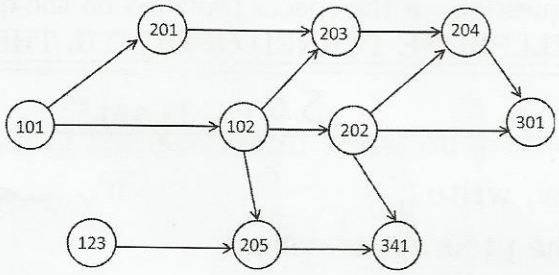
   DCGFHEBKJMNLIA

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.

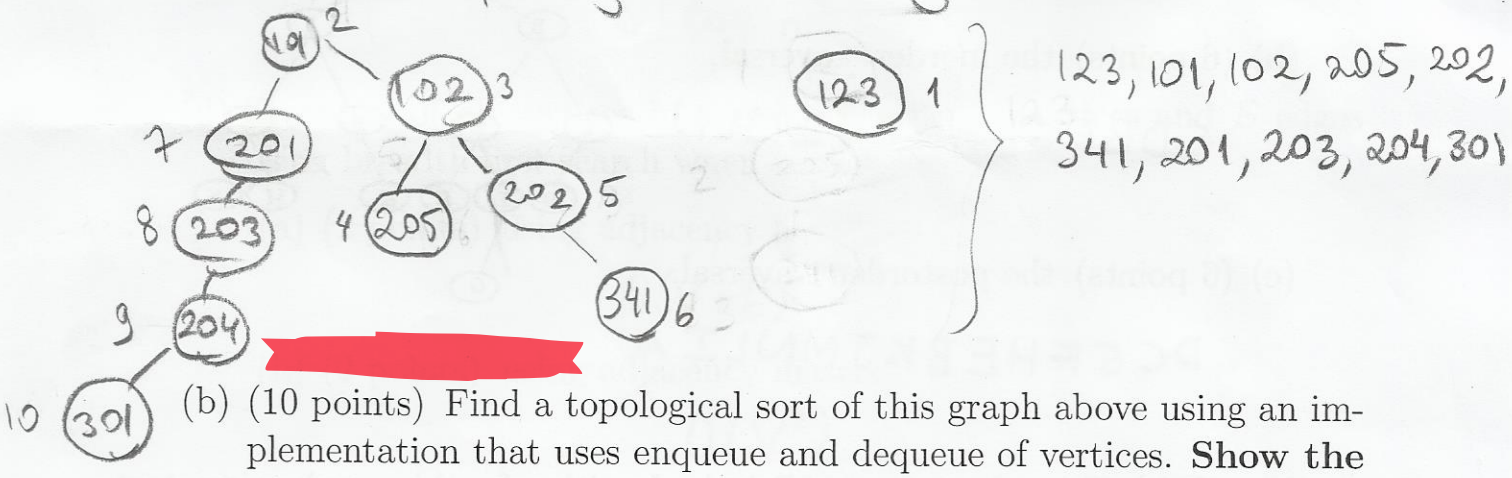
step	edge added	Weights compared
1	ab	5, 10, 11
2	bd	7, 10, 11, 12
3	ac	10, 13, 12, 16, 11, 15
4	df	12, 13, 15, 16
5	be	15, 17, 16
6	eg	15, 17



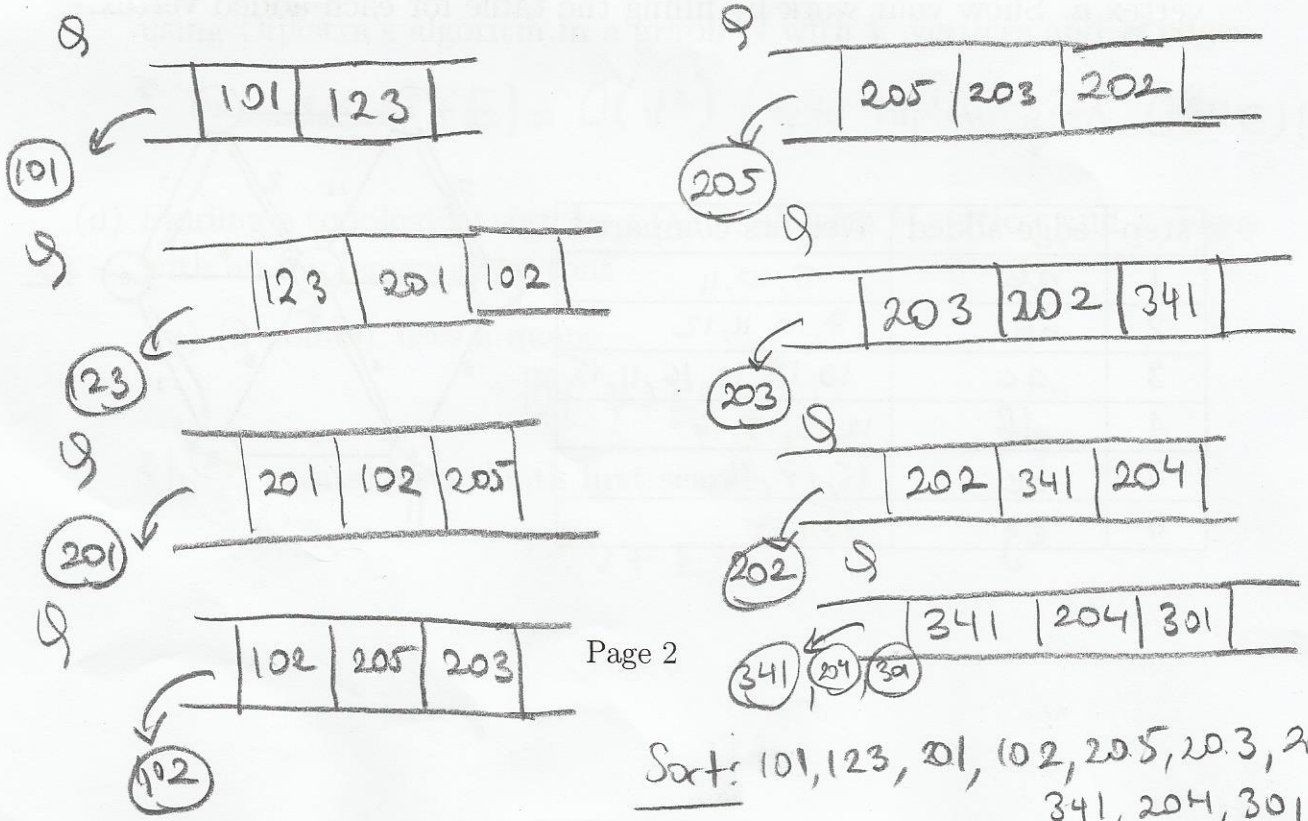
3. (a) (10 points) Find a topological sort of the graph below using depth first search. **Explain how depth first search is used to give your answer.**



Find a spanning forest using DFS:



(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.**



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

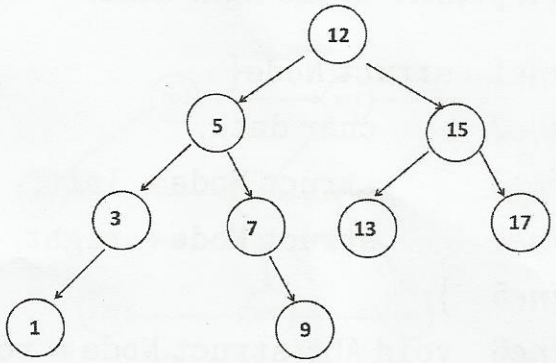
(b) (3 points) inorder traversal

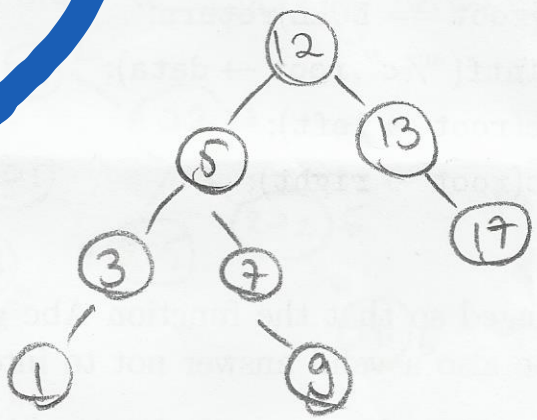
Line 9, 10 → Line 10, 9; Line 11 → Line 11

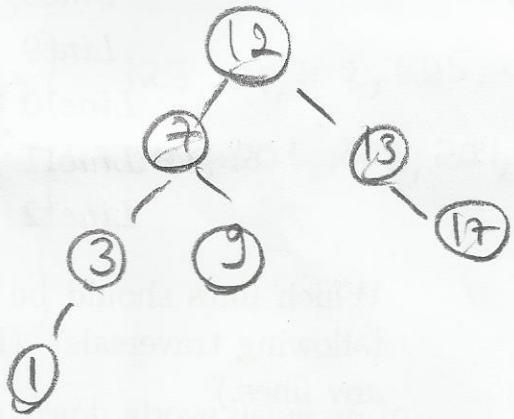
(c) (3 points) postorder traversal

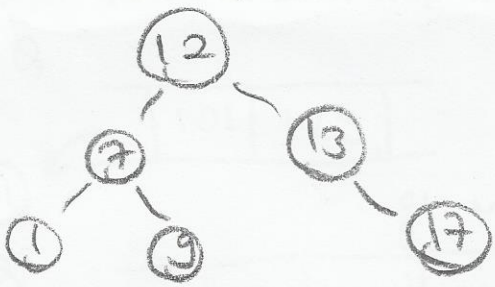
Line 9, 10 → Line 10, 11; Line 11 → Line 9

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







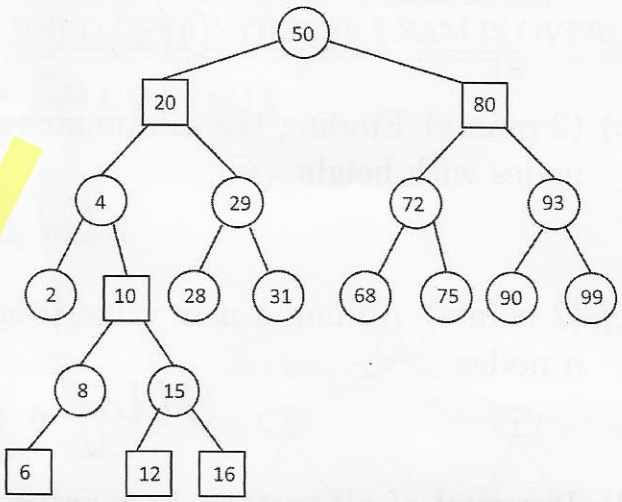


6. Consider the balanced red and black tree, where nodes in □ are red and nodes in ○ are black. Apply each operation <u>to the tree below</u> and show each step by showing the changes on the position and the color of the nodes.

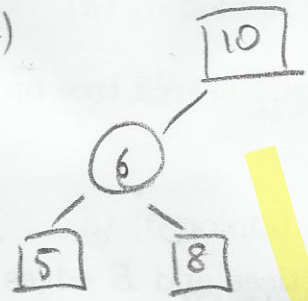
   (a) (3 points) Add 5

   (b) (4 points) Add 7

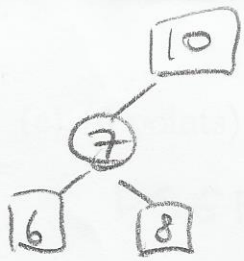
   (c) (6 points) Delete 2



a)

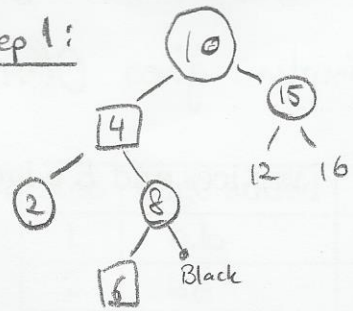


b)



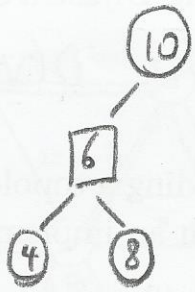
c) - Sibling of 2 is not black

Step 1:

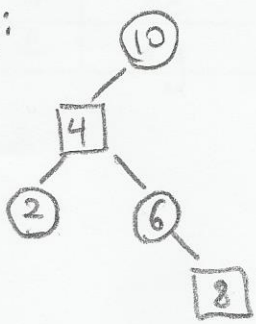


- Further away (from 2) child of 8 is black

- Rotation toward 2:



Step 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

$O(V+E)$

BBM 201 – Data Structures - Fall 2019

Final Exam
January 22, 2020

Name: _____

Student ID Number: _____ Section:_____

Problem	Points	Grade
1	20	
2	15	
3	20	
4	15	
5	19	
6	16	
Total	105	

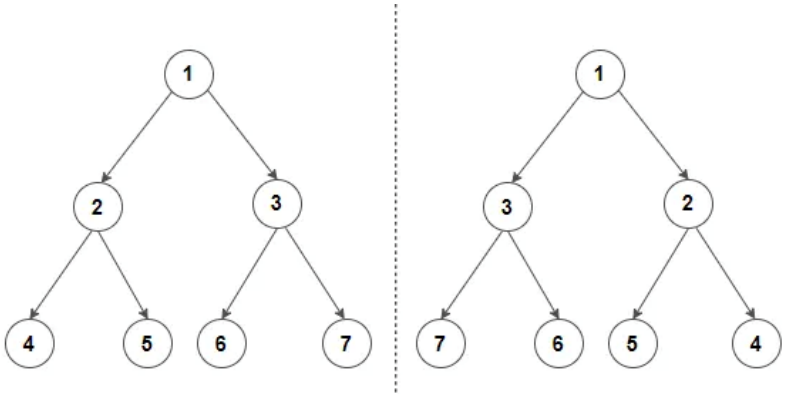
INSTRUCTIONS

- Do not open this exam booklet until you are directed to do so. Read all the instructions first.
- When the exam begins, write your name on every page of this exam booklet.
- The exam contains six multi-part problems. You have **120 minutes** to earn 105 points (5pts bonus).
- The exam booklet contains **7 pages** including this one.
- This exam is an **open book and notes exam**. You are allowed to have two pieces of **bound** books or notebooks.
- Please write your answers in the space provided on the exam paper.
- Be neat.
- Good luck!

QUESTIONS

Question 1: (18 Points)

Write a method that converts a given binary tree to its mirror as given below in the figure. The method will swap the left and right child of every node in the tree till the leaves. You can use at most one extra local variable, and no global variables. You are not allowed to do any dynamic memory allocation.




```
struct Node
{
    int data;
    struct Node* left;
    struct Node* right;
};


void mirror(struct Node* node)
{
  if (node==NULL)
    return;
  else
  {
    struct Node* temp;

    mirror(node->left);
    mirror(node->right);

    temp        = node->left;
    node->left  = node->right;
    node->right = temp;
  }
}
```

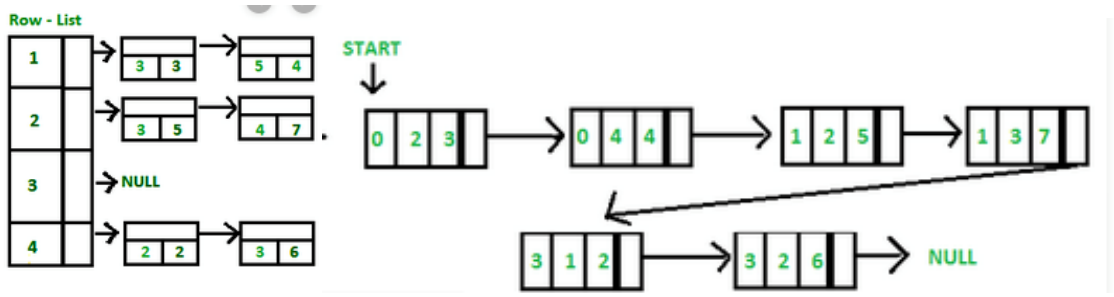
Question 2: (17 Points)

You are supposed to design an efficient data structure based on linked lists that stores a sparse matrix. For the given sparse matrix answer the below questions:

$$\begin{bmatrix} 0 & 0 & 3 & 0 & 4 \\ 0 & 0 & 5 & 7 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 2 & 6 & 0 & 0 \end{bmatrix}$$

a) Draw your data structure. Please be neat and show all the field values of each node. Any unreadable solutions will not be graded.

Any below solution or any other reasonable solution is acceptable:



b) Give the space complexity of your structure and the time complexity of finding a value in a given (x,y) location and compare it with the array based matrix structure.

O(n)

Question 1: (15 Points) Consider a tree structure with the following definitions:

- root node: a node with no parents
- branch node: a node with a parent and non-zero children
- leaf node: a node with no children

If this tree contains **m nodes**, then answer the following accordingly:

a) (2 points) What can be the maximum number of leaves on this tree?

_____m-1_____

b) (2 points) What can be the minimum number of leaves on this tree?

_____1_____

c) (2 points) What can be the maximum number of branch nodes on this tree?

_____m-2_____

d) (5 points) If each node has at most 3 children, what can be the maximum number of leaves?

_____1+floor((m-2)/3)*2+(m-2)%3_____

or

_____m-ceiling((m-1)/3)_____

or

_____floor(m/3)+ceiling(m/3)_____

e) (4 points) Given that m>1, if each branch node has at least 2 children, what can be the maximum number of branch nodes?

_____floor((m-2)/2)_____

Question 3: (15 Points) Answer each of the following questions. Keep your answers brief.

Example) Explain a real life scenario that can better be modeled with a stack rather than a queue. Explain why a stack is better than a queue in this case.

Answer: Trying to match parenthesis in an expression can better be modeled with a stack rather than a queue. The top of the stack always stores the parenthesis that needs to be matched. But the front of a queue only stores the first parenthesis ever seen and cannot be used for matching.

a) (5 Points) Explain a real life scenario that can better be modeled with a queue rather than a stack. Explain why a queue is better than a stack in this case.

Customer request arriving at a call center.

Keyboard events received by the operating system.

Clients requesting service from a server.

In all cases the early comer has priority. Queue is FIFO and respects that. Stack is LIFO and does the opposite.

b) (5 Points) Explain a real life scenario that can better be modeled with a fixed sized array rather than a linked list. Explain why the array is better than a linked list in this case.

Storing the values of a fixed number of things is better done with a fixed size array. For example, storing the grades of a class of students ordered in a specific way. This way, the grade of any student can be accessed or modified in constant time. The linked list would be much slower to do these.

c) (5 Points) Explain a real life scenario that can better be modeled with a doubly linked list rather than a binary search tree. Explain why the doubly linked list is better than a binary search tree in this case.

You want to store and process items one by one, but their order is not important. You want the operations of insertion and removal as fast as possible without a specific order. For example, you want to store and retrieve the empty structure pointers in an application rather than freeing them. Since there is no specific order among the pointers, there is no need for the more expensive tree operations.

Question # (15 points)

Consider a hashtable implementation using separate chaining of unsorted linked lists with *N* buckets and *k* items currently in the table.

a) Which of the following is true?
   1) k < N must hold
   2) k >= N must hold
   3) Neither

b) What is the average number of items in a bucket?

c) In the worst-case, how many items could be in a single bucket?

d) Given that k > N, which of the following are true?
   1) Some buckets may be empty
   2) Some buckets may be full so that they cannot receive more items
   3) Neither

e) What is the worst case running time in terms of k and N for finding the minimum value in the table ?

f) What is the worst case run time for inserting a new item in the table ?

g) If we resize the table to a table of size 2N, what is the worst case running time in terms of k and N to put all the items in the new table?

Answers:

a) Neither

b) k/N

c) k

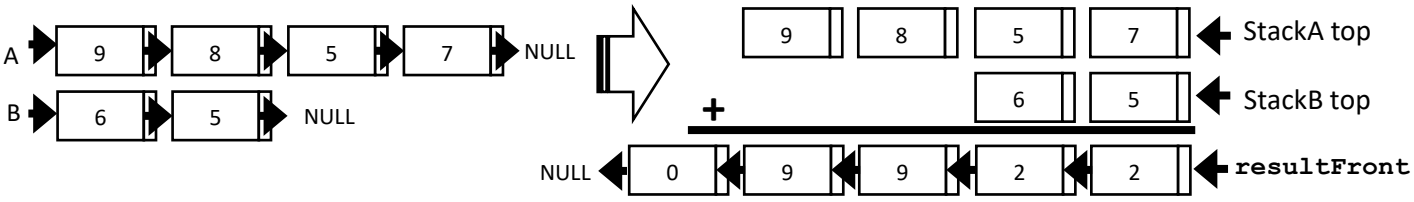
d) Some buckets may be empty

e) O(k + N). O(N) is also accepted

f) O(1)

g) O(k + N). O(N) is also accepted

Question 6: (20 points)
You are given two numbers whose digits are stored in linked lists as in the example run below. You need to add them using stacks and then store the resulting digits in a linked list. Fill in the blanks in ***int main()*** accordingly.
Hint: You need to make use of conditional expressions. Following is an example of conditional expressions: (X==Y?True:False) outputs True if X equals Y, otherwise False.
Example Run:



REQUIRED DEFINITIONS:

```
typedef struct node* node_pointer;
typedef struct node{
       int value;
       node_pointer link;
};
node_pointer resultFront;

#define MAX 1000
class Stack {
    public:
       int top;
        int a[MAX];
       Stack() { top = -1; }
       bool push(int x);
       int pop();
       bool isEmpty();
};
```

```
bool Stack::isEmpty() {
    return (top < 0);
}

int Stack::pop() {
    if (top < 0) return -1;
    else return a[top--];
}

void attach(int val, node_pointer *rear){
    node_pointer temp = new node_pointer;
    temp->value = val;
    (*rear)->link = temp;
    *rear = temp;
    (*rear)->link = NULL;
}
```

```
int main() {
    ... /* Assume that by this point the linked lists A and B have already been
    traversed and stored in StackA and StackB as in the example run above */
    node_pointer resultRear = new node_pointer;
   resultFront = resultRear;
   int e1, e2, v, carry = 0;
   while( !(StackA.isEmpty() && StackB.isEmpty()) ){
   OR while( !StackA.isEmpty() || !StackB.isEmpty() ){
   OR while( e1!=-1 || e2!=-1 ){
         e1 = StackA.pop(); e2 = StackB.pop();
         v = (e1==-1?0:e1) + (e2==-1?0:e2) + carry;
         OR v = (e1!=-1?e1:0) + (e2!=-1?e2:0) + carry;
         attach( v<10 ? v : v%10, &resultRear );
         OR attach( v%10, &resultRear );
         carry = ( v < 10 ? 0 : 1);
         OR carry = ( v >= 10 ? 1 : 0);
         OR carry = ( v / 10);
   }
   attach( carry, &resultRear );
```

```
    temp = resultFront;
    resultFront = resultFront->link;
    free(temp); return 0;
}
```

Hacettepe University
Department of Computer Engineering

BBM 201 – Data Structures
Resit Exam for Fall 2019 Semester
September 11, 2020

Problem	Points	Grade
1	15	
2	15	
3	25	
4	20	
5	25	
Total	100	

HONOR STATEMENT

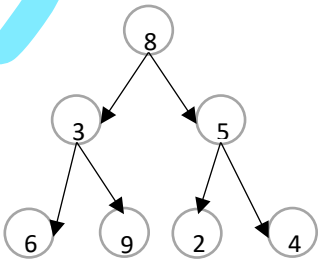
I pledge on my honor that I will not give nor receive any unauthorized help on this exam, and that all submitted work I will upload will be my own.

INSTRUCTIONS

- You have exactly **120 minutes** to finish and submit your answers to the exam.
- You must solve the questions on physical sheets of paper legibly (easy to read) with your own **handwriting**. You will upload your answers to the exam submission form by scanning or taking a picture of each answer sheet. Your answers should be complete, understandable and readable in the images you upload. Unreadable answers would be regarded as void.
- Only image files (such as JPG or PNG) or PDF files can be uploaded to the submission form. When you need to upload your answer in multiple files, you must clearly indicate their order on the answer sheets.
- For each question, you should **only** use the data structures, definitions and functions provided (or explicitly allowed) with that question. You **MUST NOT** use other data structures, definitions or functions, which are not explicitly allowed in the question.

QUESTION 1 (15)

We will use an array in order to store a full binary tree.



=>

8	3	5	6	9	2	4

Complete the given code that returns the n^{th} node in the given level. For example, find_data(3, 2) will return 9, find_data(2, 1) will return 3, and so on. You must realize the function using the definitions given below with less than or equal to 4 code lines.

```
#define ARRAY_SIZE 7
int tree[ARRAY_SIZE];
int pow (int x, int y); //assume pow function is predefined and returns x^y.

int find_data(int level, int n){

        _____________________________________________________________;
        _____________________________________________________________;
        _____________________________________________________________;
        _____________________________________________________________;

}
```

QUESTION 2 (15)

An important drawback with an array-based implementation of a stack is the resizing that must be done when the stack is full. A solution for overcoming this problem employs periodic resizing of the array by some constant amount c. That is, when the array of size k is full, a new array of size k + c is created and the elements from the original array are copied into the new one, maintaining the original order so that the stack remains intact.

Give **a complete analysis** of this solution over a sequence of **n** push operations on the stack. Assume that the stack is empty at the beginning. **Report the total cost and the average cost per push in Big-Oh notation.**

QUESTION 3 (25)

```
int magic(int low, int high) {
    return (int) low + (int) (random()%(high-low+1));
}

struct Node {
    int value = magic(0,10);
    struct Node* next;
};
```

Based on the definitions given above, fill in the blanks to realize the described function definitions exactly.
Each blank is a single line of code. Therefore, you MUST NOT write more than a single line of code in a blank.

(10) a) Create a C++ compatible function called **CreateCircularLL**(int **n**) which generates a circular linked list that consists of **n** number of nodes. The entry node to the list must be named **firstNode**. You must realize the function with less than or equal to 10 code lines.

```
Node* CreateCircularLL(int n) {
    _____________________________ = new Node;
    ____________________________________________________________;
    ____________________________________________________________;
    ____________________________________________________________;
    ____________________________________________________________;
    ____________________________________________________________;
    ____________________________________________________________;
    ____________________________________________________________;
    ____________________________________________________________;
    ____________________________________________________________;
}
```

(15) b) Create a C++ compatible function called **SearchCircularLL**(Node* **firstNode**) that finds which is the foremost element of the circular list that has the value 0. **firstNode** is the entry node of a circular list created by CreateCircularLL above. As an example, considering the list (6,1,2,0,5,0), 4th element is the foremost element with the value 0, therefore SearchCircularLL on this list would return 4. Assume that the list contains at least one element with 0 value. Also assume that **no new memory allocation** is necessary in SearchCircularLL. You must realize the function with less than or equal to 10 code lines.

```
int SearchCircularLL(Node* firstNode) {
    ____________________________________________________________;
    ____________________________________________________________;
    ____________________________________________________________;
    ____________________________________________________________;
    ____________________________________________________________;
    ____________________________________________________________;
    ____________________________________________________________;
    ____________________________________________________________;
    ____________________________________________________________;
    ____________________________________________________________;
}
```

QUESTION 4 (20)

Complete the **recursive** function **SearchValue** that searches for a given value in a regular linked list and returns the position of the value in the linked list if found, otherwise returns -1. The first item in the linked list has position 1. You must realize the function with less than or equal to 6 code lines.

```
struct Node{
        int data;
        struct Node* next;
};


int SearchValue (struct Node* firstNode, int value, int index) {

        _____________________________________________________________;
        _____________________________________________________________;
        _____________________________________________________________;
        _____________________________________________________________;
        _____________________________________________________________;
        _____________________________________________________________;
}
```

QUESTION 5 (25)

(15) a) Consider a virtual ant moves in space from point (x=0, y=0, z=0) to (x=a, y=b, z=c), where a,b and c are positive integers. **The ant moves in integer units, always in positive directions and parallel to the coordinate axis**. In other words, when it moves from (0,0,0) it can go to (1,0,0) or (0,1,0) or (0,0,1) and nowhere else. Write a **recursive** function **countWays(int a, int b, int c)** in the space given below to compute in how many **different** ways the ant can reach point (a,b,c) when it starts from (0,0,0). In other words, your function must count the number of distinct paths from (0,0,0) to (a,b,c) according to the above rules.

```
int countWays(int a, int b, int c) {

}
```

(10) b) Complete the recursive function **mod(int n, int d)** below, which returns the remainder (i.e., ***modulus)*** of $n \div d$. Please note that this does not imply integer division of n by d. And you **must *not*** use division (/), multiplication (*), or modulus (%) operators, or any other helper functions. You should also assume that $n \geq 0$ and $d \geq 1$.

```
int mod(int n, int d) {

}
```

QUESTION 3

a)

```
Node* createCircle(int nodes) {
   Node* firstNode = new Node();
   int count = 1;
   Node* nextNode = firstNode;
   while (count < nodes) {
      nextNode->next = new Node();
      nextNode = nextNode->next;
      count++;
   }
   nextNode->next=firstNode;
   return firstNode;
}
```

b)

```
Node* createCircle(int nodes) {
   Node* firstNode = new Node();
   int count = 1;
   Node* nextNode = firstNode;
   while (count < nodes) {
      nextNode->next = new Node();
      nextNode = nextNode->next;
      count++;
   }
   nextNode->next=firstNode;
   return firstNode;
}
```

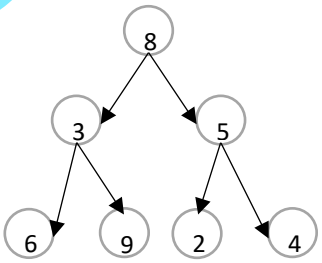
QUESTION 4

Complete the function **SearchItem** that searches for a given value in a regular linked list and returns the position of the value in the linked list if found, otherwise returns -1. The first item in the linked list has position 1.

```
struct Node{
       int data;
       struct Node* next;
};


int  SearchItem(struct Node* head, int value, int index)
{


       if(head==null)
              return -1;
       else if(head->data==value)
              return index+1;
       else
              return SearchItem(head->next,value,index+1)
}
```

QUESTION 1

We will use an array in order to store a full binary tree.



Complete the given code that returns the n^{th} node in the given level. For example, find_data(3, 2) will return 9, find_data(2, 1) will return 3, and so on. You must realize the function using the definitions given below with less than or equal to 4 code lines.

```
#define ARRAY_SIZE 7
int tree[ARRAY_SIZE];
int pow (int x, int y); //assume pow function is predefined and returns x^y.

int find_data(int level, int n){

return tree[pow(2, level-1)-1+n-1];

}
```

QUESTION 2

(5 points) One problem with an array-based implementation of a stack is the resizing we must do when the stack fills. A scheme for overcoming this drawback employs periodic resizing of the array by some constant amount c. That is, when the array of size k fills, we create a new array of size $k + c$ and copy the elements from the original array into the new one, maintaining the original order so that the stack is not corrupted. Give a complete analysis of this scheme over a sequence of n `push` operations on the stack. Assume that the stack is empty to start. Please report your answer as an average cost per `push`.

Solution: over n pushes there are n/c copying events, and that in the ith copy event, $i \cdot c$ data items are copied from the old full array to the new one. The following sum reflects the total number of copies:

$$\sum_{i=0}^{\frac{n}{c}} i \cdot c = c \sum_{i=0}^{\frac{n}{c}} i = \frac{c}{2} \cdot \frac{n}{c}(\frac{n}{c} + 1) = O(n^2)$$

Over n pushes the average cost per push is thus $O(n)$.

Rubric: 3 points for analysis, 2 points for correct answer. Two of three analysis points were awarded for any answer in which the process was decomposed appropriately into the number of copy events and the size of each, even if the values used were incorrect. Forgetting to report the average was a 1 point penalty.

QUESTION 6

a) Consider a virtual ant moves in space from point (x=0, y=0, z=0) to (x=a, y=b, z=c), where a,b and c are positive integers. **The ant moves in integer units, always in positive directions and parallel to the coordinate axis**. In other words, when it moves from (0,0,0) it can go to (1,0,0) or (0,1,0) or (0,0,1) and nowhere else. Write a **recursive** function to compute in how many **different** ways the ant can reach point (a,b,c) when it starts from (0,0,0). In other words, your function must count the number of distinct paths from (0,0,0) to (a,b,c) according to the above rules.

```
int countWays(int a, int b, int c) {
        if (x + y + z == 1)
                return 1;
        else {
                int xPath = 0, yPath = 0, zPath = 0;
                if (x > 0)
                        xPath = countWays(x - 1, y, z);
                if (y > 0)
                        yPath = countWays(x, y - 1, z);
                if (z > 0)
                        zPath = countWays(x, y, z - 1);
                return xPath + yPath + zPath;
        }
}
```

b)

```
int mod(int n, int d) {
```

```
    if (n < d)
        return n;
    else
        return mod(n-d, d);
```

```
}
```

Q1 Academic Honesty

0 Points

It is a violation of the Academic Integrity Code to look at any reference material other than your textbook and lecture notes, or to give inappropriate help to someone or to receive unauthorized aid by someone in person or electronically via messaging apps such as WhatsApp. Academic Integrity is expected of all students of Hacettepe University at all times, whether in the presence or absence of members of the faculty. Do NOT sign nor take this exam if you do not agree with the honor code.

Understanding this, I declare I shall not give, use or receive unauthorized aid in this examination.

Signature (Specify your name and surname as your signature)

While answering the following questions, please consider the content that we discussed in our lectures unless stated otherwise.

Q2

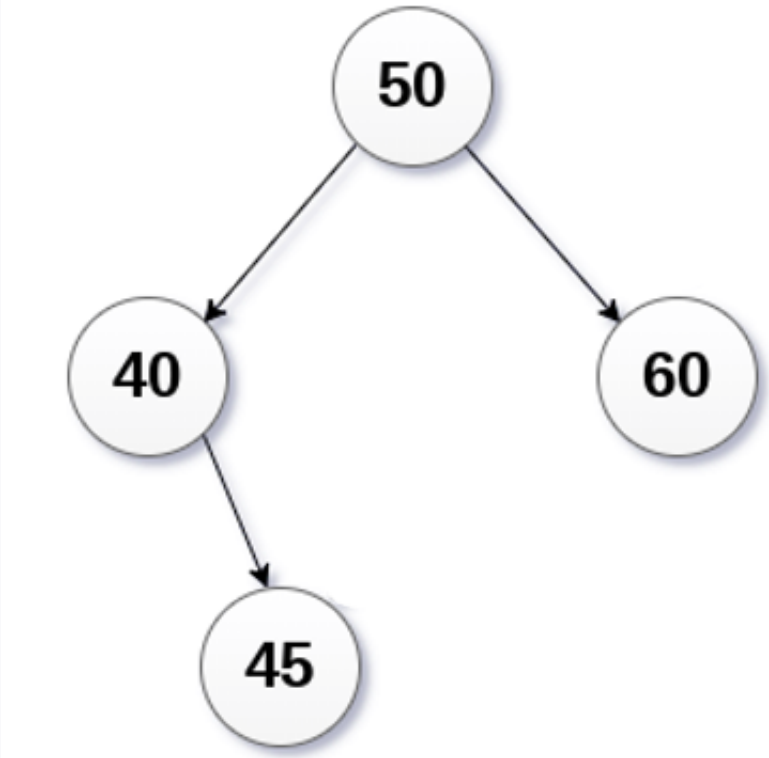
12 Points

For each of the below AVL trees answer the question below:

Q2.1

4 Points

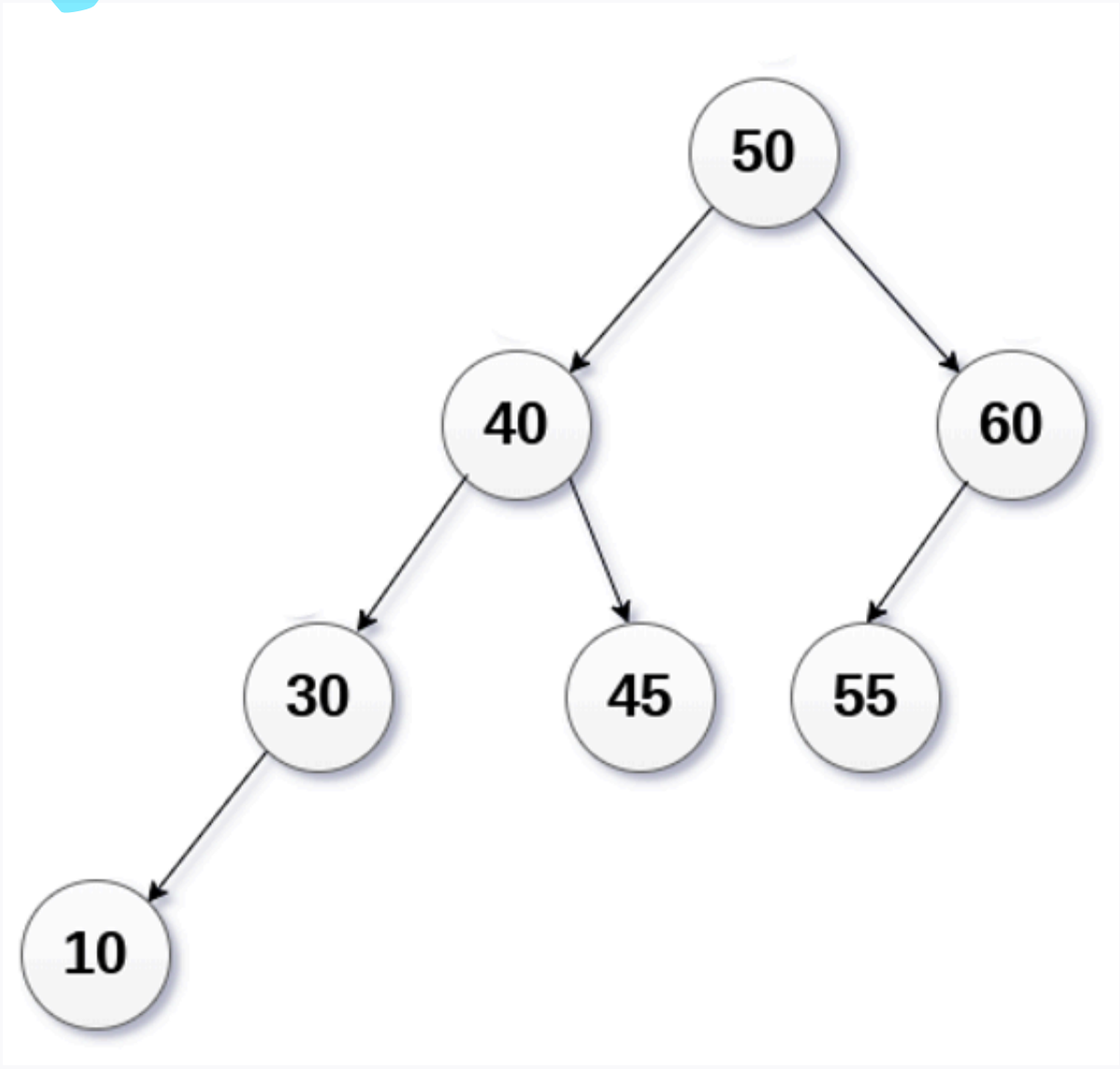
a)After deletion of the node 60 from the AVL tree shown below, what will be the root of the new tree?



45

Q2.2
4 Points

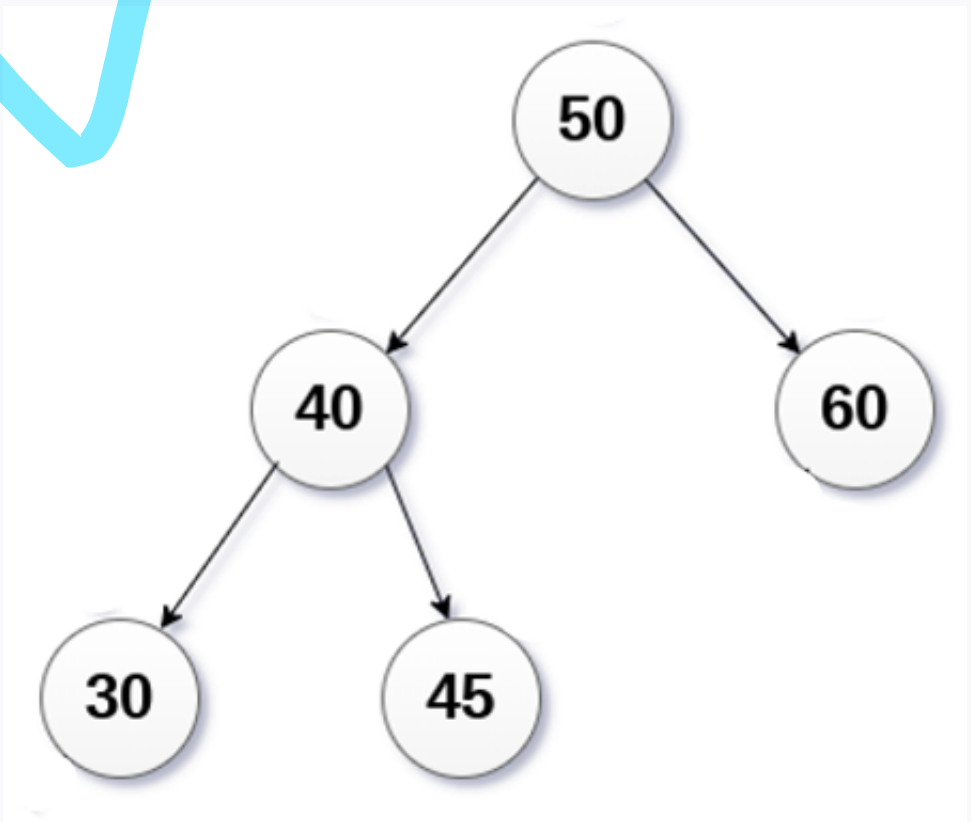
b)After deletion of the node 55 from the AVL tree shown below, what will be the right child of 40 in the new tree?



50

Q2.3
4 Points

c)After deletion of the node 60 from the AVL tree shown below, what will be the parent of 45 in the new tree?



50

Q3
12 Points

We would like to store 10 integer values. Answer below questions.

Q3.1
4 Points

If we use an array based linked list where array size is 15, what will be the size of the total structure in bytes? (do NOT include the first and free index holders) Assume integers and pointers are 4 bytes.

120

Q3.2
4 Points

If I use a complete tree structure and use minimum space representation possible, what will be the size of the total structure in bytes? Assume integers and pointers are 4 bytes.

120

Q3.3
4 Points

If I use a singly linked list, what will be the size of the total structure in bytes? (do NOT include the head pointer) Assume integers and pointers are 4 bytes.

80

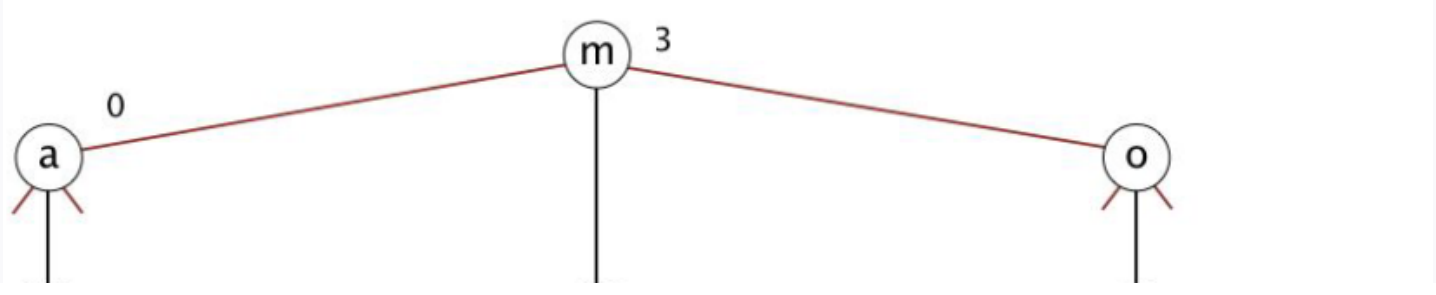
Q4
11 Points

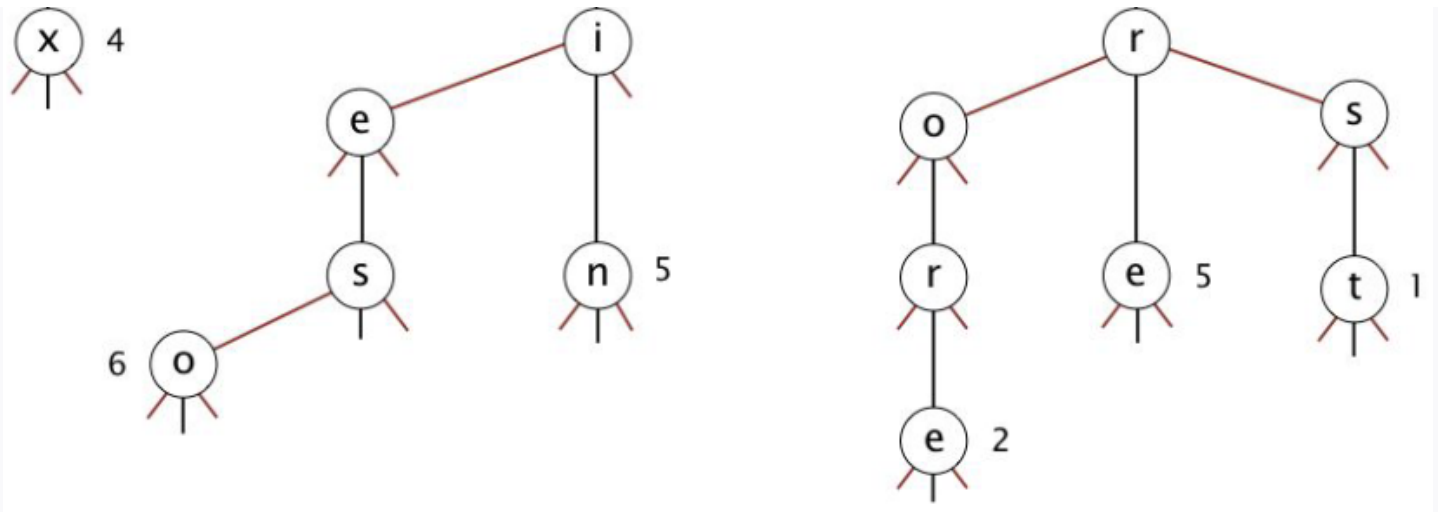
A Ternary Search Trie (TST) is a special trie data structure where the child nodes of a standard trie are ordered as follows:

- The left pointer points to the node whose value is less than the value in the current node.
- The middle pointer points to the node whose value is equal to the value in the current node.
- The right pointer points to the node whose value is greater than the value in the current node.

Search in a TST follows the links corresponding to each character in the key such that

- If less, take the left link; if greater, take the right link.
- If equal, take the middle link and move to the next key character.





A TST is shown above. Values are indicated with a number next to the node to represent a string symbol table.

Q4.1
5 Points

Choose all the strings that the TST contains. (no partial credit)

- [x] a
- [x] ax
- [x] ore
- [x] oore
- [x] ost

Q4.2
6 Points

Choose all the strings that the TST contains. (no partial credit)

- [x] m
- [ ] max
- [x] meo
- [x] min
- [ ] more
- [ ] most

Q5
10 Points

Build a hash table using separate chaining method. For hash function use h(x)= x mod 9 for a table size of 9.

key values: 19 13 1 12 22 24 7 16 35 10

Once you have the hash table, For the access pattern: 5 24 7 25 35 21 1 19

Give the probe time for each access pattern below (leave a single space in between). Note: accessing the index array is counted as one probe.

Give the average probe time. (use dot (.) not comma (,) for floating point numbers)

Q6
25 Points

For each of the following three-string symbol table implementations, indicate how characters and strings are stored by picking the best match from among three choices:

For the definition of a Ternary Search Trie (TST) refer to Q4

Q6.1
5 Points

Neither characters nor strings are stored explicitly

- ○ Hash table (linear probing)
- ◉ R-way trie
- ○ TST

Q6.2
5 Points

Characters are stored explicitly but strings are not

- ○ Hash table (linear probing)
- ○ R-way trie
- ◉ TST

Q6.3
5 Points

Strings (and characters) are stored explicitly

- ◉ Hash table (linear probing)
- ○ R-way trie
- ○ TST

Q6.4
5 Points

Suppose a million random strings, each of length 10, over a 256-character alphabet are inserted into a symbol table. Which one would consume the most space?

- ○ Hash table (linear probing)
- ◉ R-way trie
- ○ TST

Q6.5
5 Points

As before suppose a million random strings, each of length 10, over a 256-character alphabet are inserted into a symbol table. Which one would consume the least space?

- ◉ Hash table (linear probing)
- ○ R-way trie
- ○ TST

Q7
10 Points

A d-max-heap is like an ordinary binary max-heap, except that nodes have d children instead of 2.

Q7.1
5 Points

Describe how a d-max-heap can be represented in an array A[1. . . n]. In particular, for the internal (non-leaf) node of the d-max-heap stored in any location A[i], consider the positions in A[] hold its child nodes. Assuming the indices start at 0, select the ones that are correct. (no partial credit)

- ☐ *1* st child is at A[d*i−d+1]
- ☑ *2* nd child is at A[d*(i-1)+1]
- ☐ *d* th child is at A[d*i]
- ☑ *k* th child is at A[d*k+i]

Q7.2
5 Points

What is the height of the heap to be the number of nodes on the longest path from the root to a leaf. In terms of n and d, what is the height of a d-max-heap of n elements?

- ○ $\log_n d$
- ○ $\log_2 n$
- ◉ $\log_d n$
- ○ $\ln n$

Q8 Graph

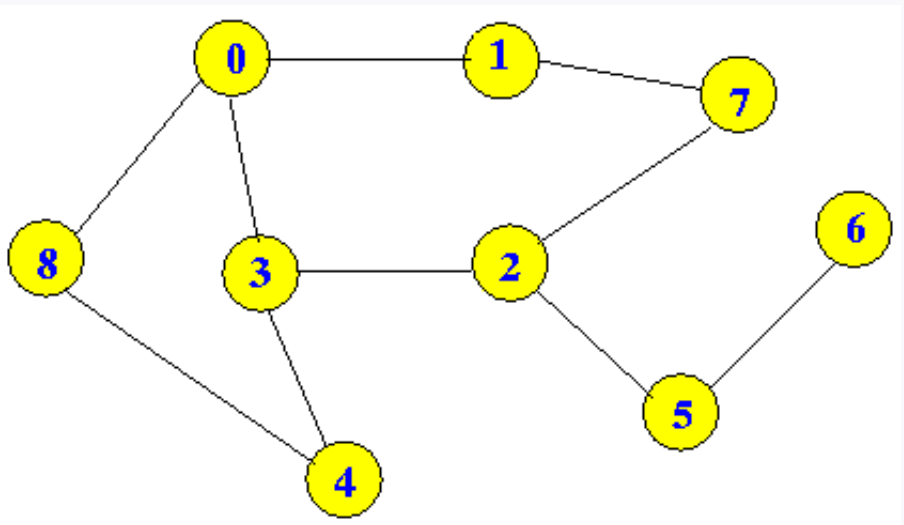
10 Points

Suppose you have a graph and any pair of vertices are chosen as start and goal. There are many paths from the start vertex to the goal vertex. You also know that they tend to be rather long. Suppose that you want to implement a program that searches for a path and returns the first one it can find. You have no need for finding the optimal path in any sense, just any path will do. Would you want to use BFS or DFS as the basis for your program? Justify your answer in at most two sentences.

Q9

10 Points

Given the graph below, if the adjacent vertices of a vertex are kept in a priority queue using the min-heap structure, what will be the visit order of the below graph when doing BFS starting from 0 vertex?



Give your answer below, putting a space in each number

0 1 3 8 7 2 4 5 6

Final Exam

● GRADED

STUDENT

TOTAL POINTS

71 / 100 pts

QUESTION 1

Academic Honesty — **0** / 0 pts

QUESTION 2

(no title) — **12** / 12 pts

- 2.1 (no title) — **4** / 4 pts
- 2.2 (no title) — **4** / 4 pts
- 2.3 (no title) — **4** / 4 pts

QUESTION 3

(no title) — **8** / 12 pts

- 3.1 (no title) — **4** / 4 pts
- 3.2 (no title) — **0** / 4 pts
- 3.3 (no title) — **4** / 4 pts

QUESTION 4

(no title) — **11** / 11 pts

- 4.1 (no title) — **5** / 5 pts
- 4.2 (no title) — **6** / 6 pts

QUESTION 5

(no title) — **0** / 10 pts

QUESTION 6

(no title) — **25** / 25 pts

- 6.1 (no title) — **5** / 5 pts
- 6.2 (no title) — **5** / 5 pts
- 6.3 (no title) — **5** / 5 pts
- 6.4 (no title) — **5** / 5 pts
- 6.5 (no title) — **5** / 5 pts

QUESTION 7

(no title) — **5** / 10 pts

- 7.1 (no title) — **0** / 5 pts
- 7.2 (no title) — **5** / 5 pts

QUESTION 8

Graph — **0** / 10 pts

QUESTION 9

(no title) — **10** / 10 pts

Hacettepe University
Department of Computer Engineering

BBM 201 – Data Structures
Three-Course Exam - October 6, 2020

Problem	Points	Grade
1	15	
2	20	
3	25	
4	20	
5	20	
Total	100	

HONOR STATEMENT

I pledge on my honor that I will not give nor receive any unauthorized help on this exam, and that all submitted work I will upload will be my own.

INSTRUCTIONS

- You have exactly **120 minutes** to finish and submit your answers to the exam.
- This is a regular written exam for which you will turn in your **handwritten** answers using the provided Google Form link.
- This is an open-book exam. You may make use of books, notes, and other online or offline resources. However, **your answers must be your own**. In other words, **your answers MUST NOT be copied from some other resource/answer**. In case your submitted answers are substantially similar to those submitted by other students or those provided by other resources, it will invalidate your exam and trigger a disciplinary investigation.
- You must solve the questions on physical sheets of paper legibly (**easy to read**) with your own **handwriting**. You will upload your answers to the exam submission form by scanning or taking a picture of each answer sheet. Your answers should be complete, understandable and readable in the images you upload. **Typed submissions or unreadable submissions would be regarded as void**.
- Only image files (such as JPG or PNG) can be uploaded to the submission form. When you need to upload your answer in multiple files, you must clearly indicate their order on the answer sheets.
- For each question, you should **only** use the data structures, definitions and functions provided (or explicitly allowed) with that question. You **MUST NOT** use other data structures, definitions or functions, which are not explicitly allowed in the question.

QUESTION 1

(15 points)

(a) **(5 points)** Find what RecursiveFunc (given below) does **in general**.

Finds the maximum of elements in a

(b) **(10 points)** Print the output of the following program.

```
int RecursiveFunc (int array[], int index, int n)
{
     int val1, val2;
     if ( n==1 )
          return array[index];
     val1 = RecursiveFunc (array, index, n/2);
     val2 = RecursiveFunc (array, index+(n/2), n-(n/2));
     if (val1 > val2)
          return val1;
     else
          return val2;
}
...
int a[8] = { 1,2,10,15,16,4,8,2 };
printf( "value is % d \n", RecursiveFunc(a,0,8));
```

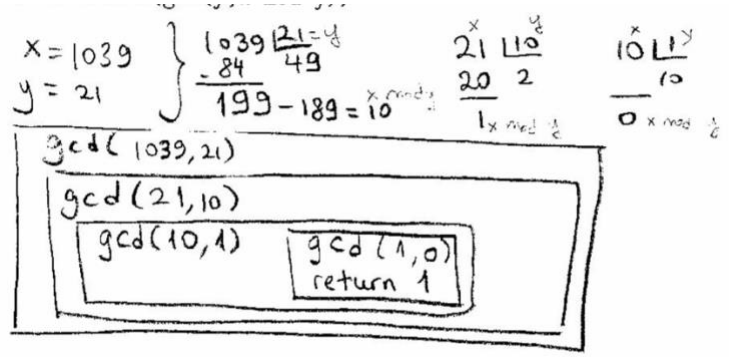
value is 16

QUESTION 2
(20 points)

(a)

(10 points) Below is a pseudocode of the Euclidean algorithm that calculates the greatest common divisor (gcd) of any given two integers. Write all recursive calls made to calculate `gcd(1039, 21)` by using a *box diagram*.

```
algorithm gcd(x,y)
  if y = 0
    then return(x)
  else return(gcd(y,x mod y))
```



(b)

(10 points) Write the complexity of the running time of the following code fragment using big-Oh notation. Show your work.

```
int count = N, i = 0, j = 0;
for (i = 0; i<count; i++)
      if(i%2==0)
            j++;
      else
            j--;
for (i = count; i > 0; i = i/2)
      for (j = 0; j < i; j++)
            count ++;
```

$$\left\lceil \frac{N}{2} \right\rceil + \left\lceil \frac{N}{2} \right\rceil + N + \frac{N}{2} + \frac{N}{4} + \cdots + \frac{N}{2^k} =$$

$$k = \log_2 N \qquad = O(N)$$

QUESTION 3
(25 points)

How many array accesses do the following code fragments make as a function of N?

(a) (8 points)

```
int count = 0;
for (int i = 0; i < N; i++)
     for (int j = i+1; j < N; j++)
          if (a[i] + a[j] == 0)
               count++;
```

$$2\cdot\left[(N-1)+(N-2)+\cdots+1\right]=2\binom{N}{2}=\frac{N(N-1)}{2}\cdot 2$$

(b) (8 points)

```
int count = 0;
for (int i = 0; i < N; i++)
     for (int j = i+1; j < N; j = 2*j)
          if (a[i] + a[j] == 0)
               count++;
```

1

N times

$$2\left[\log_2(N-1)+\log_2(N-1)+\cdots+\log_2(N-1)\right]$$

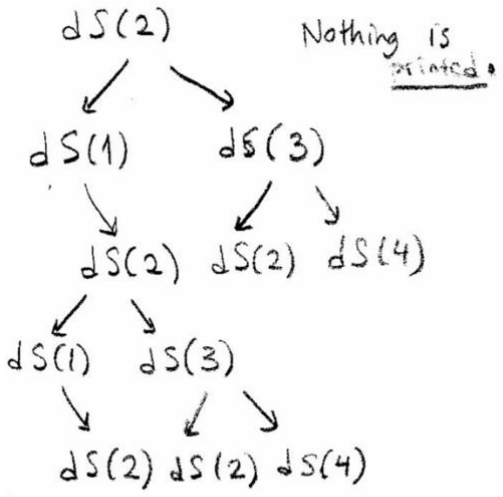
($i=0$ to $i=N-1$; $i=1$)

$$\sim 2N\log_2 N$$

(c)

(9 points) What will be the output when `doSomething(2)` is called?

```
void doSomething(int value)
{
     if(0 < value && value < 10)
     {
          doSomething(value - 1);
          doSomething(value + 1);
          printf(" %d", value);
     }
}
```



Nothing is printed.

Never Stops

QUESTION 4
(20 points)

(a) (10 points) A palindromic word is a sequence of characters that reads the same backward and forward. For example; *pepper*, *refer*, *kayak* are palindromic words. The method IsPalindrome(char[] str) checks if a given string is palindrome by using a stack. According to the given prototypes of stack methods, please fill in the gaps in the code.

```
int stack[10];
int top=-1;
void push(char); //pushes the given char on the stack.
char pop();      //pops a char from the stack.

void IsPalindrome(char str[]){
      int len = strlen(str), i, count=0;
      len = strlen(str);

      for (i = 0; i < len; i++)
         push(str[i]);

     for (i = 0; i < len; i++)
         if (str[i] == pop())
             count++;

     if (count==len)
          printf("\n%s is a Palindrome string\n", str);
      else
          printf("\n%s is not a palindrome string\n", str);
}
```

(b) (10 points) Postfix is preferable than infix for computers because of having no checks for parenthesis and no check for operator precedence rules. Prefix has also the same properties as postfix for having no parenthesis and precedence check. Give a reason why postfix is more preferable than prefix for evaluation of expressions by the computers?

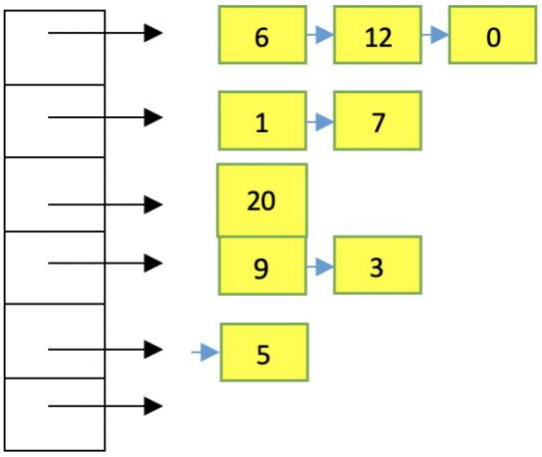
We can evaluate a postfix notation by using only one stack, whereas in prefix notation we have to use at least two stacks for the evaluation. In addition, we need to traverse to the end of the prefix notation and then we need to calculate the expression backwards.

QUESTION 5
(20 points)

```
#define TABLE_SIZE 6

typedef struct list *list_pointer;
typedef struct list {
    int key;
    list_pointer link;
}list;
list_pointer hash_table[TABLE_SIZE];

int hash_function(int key){
      return key % TABLE_SIZE;
}
void insert(int key)
{
      int hash_value = hash_function(key);
      list_pointer ptr, trail=NULL, lead=hash_table[hash_value];
      for (; lead; trail=lead, lead=lead->link)
            if (lead->key==key) {
                  printf("The key is in the table\n");
                  exit(1);
            }
       ptr = (list_pointer) malloc(sizeof(list));

       ptr->key = key;
       ptr->link = NULL;
       if (trail) trail->link = ptr;
       else hash_table[hash_value]= ptr;
}
```

According to the above hashtable implementation, fill in the below figure after the following method calls are applied:

insert(9); insert(20); insert(6);insert(12); insert(3); insert(5); insert(0); insert(1); insert(7);



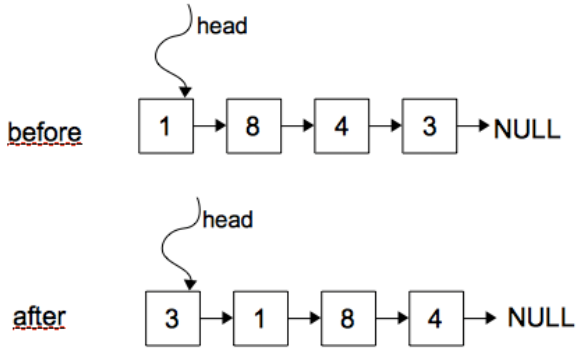
1. 1. What is the output of the following method if it is called for a linked list with items 5-3-6-8-2?

```
void recMethod(node* item){

        if(item->next!=NULL){
                printf("%d ",item->x);
                recFun(item->next);
                printf("%d ",item->next->x);
        }
}
```

2. Write a **recursive** method to free a given linked list.

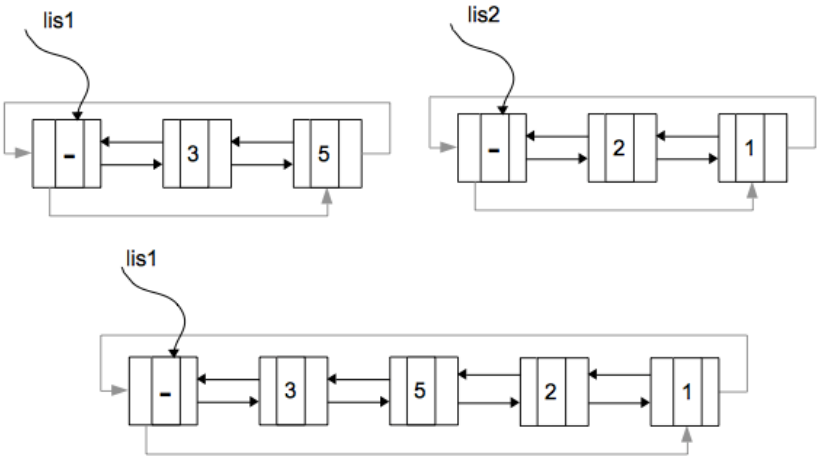
```
struct node{
    int data;
    struct node* next;
};

void freeList(struct node* head){









}
```

3. The insertBeginning(node* head) method inserts the last node of the given linked list to the beginning of the list (see the figure below). Please find the logical error(s) and correct them on the code given below:




```
typedef struct node{
  int x;
  struct node *next;
}node;

void insertBeginning(node* head){
    node *pre, *old_head=head;
    for(;head->next;pre=head,head=head->next);
    pre->next=NULL;
    head->next=old_head;
}


int main()
{
   node* head;
          ……
   insertBeginning(head);
          ……

}
```

4. Complete the following method that concatenates two circular doubly linked lists by removing the header of the second linked list (see the figures below).




```
typedef struct node{
  int x;
  struct node *pre;  //previous link
  struct node *fol;  //following link
}node;

node* concatLists(node* list1, node* list2){

if(list2->pre==list2){  // or if(list2->fol==list2)
   free(list2);
   return list1;
}

list1->pre=list2->pre;
list2->pre->fol=list1;
list1->pre->fol=list2->fol;
list2->fol->pre=list1->pre;


}
```

5. Please convert the following prefix expression into infix and postfix notations:

   /*-6/7+5/3+1*24-15-28

   a) Infix notation:

   b) Postfix notation:

6. Draw the linked list implementation of the given sparse matrix according to the given structure definitions. You do not need to write the tag fields.

```
typedef enum {head, entry} tagfield;
typedef struct matrix_node * matrix_pointer;
typedef struct entry_node{
    int col;
    int row;
    int value;
};
typedef struct matrix_node{
    matrix_pointer down;
    matrix_pointer right;
    tagfield tag;
    union {
        matrix_pointer next;
        entry_node entry;
    }u;
};
```

0	1	0	0
2	0	1	0
0	0	0	1

1. Write the **recursive** function that converts an integer into its binary representation and returns the number in the binary form.

```
int  binaryConversion(int number)
{
     if (num == 0)
        return 0;
     else
        return (num%2)+10*binary_conversion(num/2);
}
```

2. Write a **recursive** function that reverses every alternate k nodes (where k is an input to the function) in a singly linked list. Using global variables or adding more parameters to the function are forbidden.

Example:

Inputs: 1->2->3->4->5->6->7->8->9->NULL and k=3
Outputs: 3->2->1->4->5->6->9->8->7->NULL

```
struct node
{
        int data;
        struct node* next;
}

struct node *kAltReverse(struct node *head, int k)
{

struct node *kAltReverse(struct node *head, int k)
{
  struct node* current = head;
  struct node* next;
  struct node* prev = NULL;
  int count = 0;

  /*1) reverse first k nodes of the linked list */
  while (current != NULL && count < k)
  {
    next  = current->next;
    current->next = prev;
    prev = current;
    current = next;
    count++;
  }

  /* 2) Now head points to the kth node.  So change next
     of head to (k+1)th node*/
  if(head != NULL)
    head->next = current;

  /* 3) We do not want to reverse next k nodes. So move the current pointer to skip
next k nodes */
  count = 0;
  while(count < k-1 && current != NULL )
  {
    current = current->next;
    count++;
  }

  /* 4) Recursively call for the list starting from current->next.And make rest of the list
as next of first node */
  if(current !=  NULL)
    current->next = kAltReverse(current->next, k);

  /* 5) prev is new head of the input list */
  return prev;
}
```

3. Write a function that evaluates a given prefix expression by using the given stack and the methods.

```
int stack[10];
int top=-1;
void push(int val)
{
   stack[++top]=val;
}
int pop()
{
   return(stack[top--]);
}

int evaluatePrefix(char prefix[10])
{
     int len,val,i,opr1,opr2,res;
          len=strlen(prefix);
          for(i=len-1;i>=0;i--)
          {
            switch(get_type(prefix[i]))
            {
                case 0:
                val=prefix[i]-'0';
                push(val);
                break;

           case 1: opr1=pop();
           opr2=pop();
           switch(prefix[i])
           {
                case '+': res=opr1+opr2;
                break;
                case '-': res=opr1-opr2;
                break;
                case '*': res=opr1*opr2;
                break;
                case '/': res=opr1/opr2;
                break;
           }
           push(res);

}

int get_type(char c)
{
if(c=='+'||c=='-'||c=='*'||c=='/')
return 1;
else
return 0;
```