EASTERN MEDITERRANEAN UNIVERSITY
Computer Engineering Department

CMPE-231 DATA STRUCTURES

FINAL EXAMINATION

24 June 2022

Duration: 100 minutes

- **Name, Surname** __SOLUTION KEY_____
- **Student ID #** _____

Grading:

PART I.	*/25*
PART II.	*/25*
PART II.	*/51*
Total	*/101*

- Please, check your exam sheet and make sure that it contains **12 pages**. In case of **any missing pages**, **inform the invigilator** IMMEDIATELY !
- Use of calculators or any electric devices is not allowed.

C programming language: Operators

OPERATORS	ASSOCIATIVITY
( ) [ ] -> .	Left to right
! ++ -- + - * & (type) sizeof	Right to left (Unary)
* / %	Left to right
+ -	Left to right
< <= > >=	Left to right
== !=	Left to right
&&	Left to right
||	Left to right
?:	Right to left
= += -= *= /= %=	Right to left
,	Left to right

PART I) (25 points) Answer the following multiple choice questions.

1) Which of the following operations is performed more efficiently by doubly linked list than by singly linked list?

(A) Deleting a node whose location is given
(B) Searching of an unsorted list for a given item
(C) Inserting a node after the node with given location
(D) Traversing a list to process each node

A

2) A characteristic of the data that the binary search uses but the linear search ignores is

(A) Type of elements of the list
(B) Length of the list
(C) Maximum value in list
(D) Order of the elements of the list

D

3) Which data structure is used for implementing recursion?

(A) Queue **(B)** Arrays **(C)** Stack **(D)** List

C

4) In the binary search algorithm, the average number of comparisons required for searching an element in a list of n numbers scales as

(A) $\log_2 n$ **(B)** $n / 2$ **(C)** n **(D)** $n - 1$

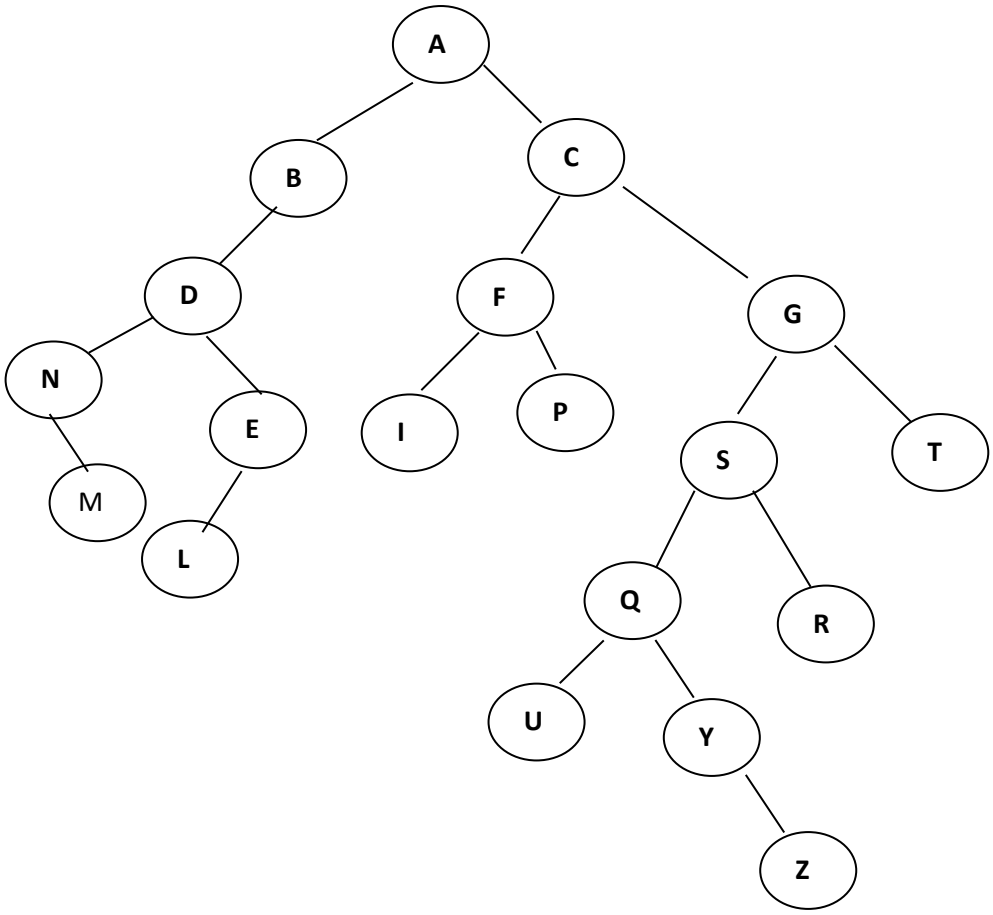
A

5) In the Quicksort algorithm, the average number of comparisons required for sorting a list of n numbers scales as

(A) $\log_2 n$ **(B)** $n \log_2 n$ **(C)** n **(D)** n^2

B

PART II) (25 points) Consider the binary search tree below, where the item values are not shown explicitly but instead variables that store item values are shown. No two variables have the same value.



Assuming that the tree was constructed for sorting the items (integers) in **descending (decreasing)** order (Hint: C > Y)

1- which of the following is correct?

a) Item **I** is the smallest
b) Item **N** is the largest
c) **Q+S < Z+T**
d) **P > B**

Ans:

b

2- which of the following is **not** correct?

a) **M+N > E+L**
b) **I+S > F+G**
c) **B+P > N+A**
d) **D+F > C+Q**

Ans:

c

3- which of the following is correct?

a) 5 items have values smaller than the value of **Y**
b) 2 items have values smaller than the value of **D**
c) 3 items have values greater than the value of **C**
d) None of the above

Ans:

a

4- which of the following is **not** correct?

a) in-order traverse of the subtree with the root **D** is **NMDLE**
b) pre-order traverse of the subtree with the root **C** is **CFIPGSQUYZRT**
c) post-order traverse of the subtree with the root **B** is **NMELDB**
d) It may be that **N > C+P**

Ans:

c

5- which of the following is **not** correct?

a) after the deletion of node **C** the pre-order traverse of the tree can be ..(*assume this part is correct*).. **PFIGS**..(*assume this part is correct*)..

b) after the deletion of node **C** the pre-order traverse of the tree can be ..(*assume this part is correct*)..**UFIPGS**..(*assume this part is correct*)..

c) two different node deletion algorithms must result in the same in-order traverse

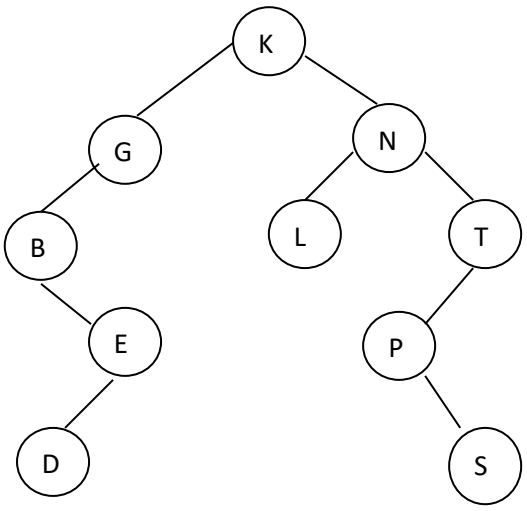
d) after the deletion of node **C** the post-order traverse of the tree can be ..(*assume this part is correct*).. I**PFGUZ**..(*assume this part is correct*)..

Ans:

d

PART III) (51 points)

1)(10 points) Consider the following binary search tree:



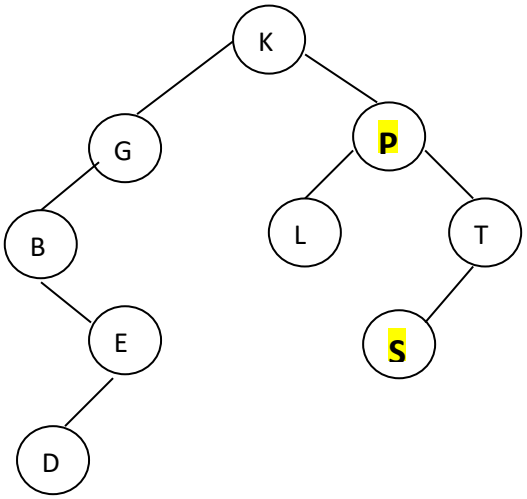
a.) Give the **preorder** traverse of the tree

KGBEDNLTPS

b.) Give the **inorder** traverse of the tree

BDEGKLNPST

c.) Draw the binary search tree when the node with the value **N** is deleted subject to that the node L remains at the same position.



2)(7 points) Consider the array implementation of the *circular queue* defined by the following code:

```
#define MAXQUEUE 4
struct queue
{
     char items[MAXQUEUE];
     int front, rear;
} q;

q.front=q.rear=MAXQUEUE-1;
```

Assume that we insert the item '**A**' into the queue **q**. The queue then appears as follows:

0	1	2	3
'A'			

q.front = 3, q.rear = 0

a) *Show the contents* of the queue and the values of **q.front** and **q.rear** after the sequence of following operations:

1.) Insert **'D'** to the queue that already contains one item **'A'** as above.
2.) Insert **'C'**.
3.) Remove an item.

Your answer:

0	1	2	3
	'D'	'C'	

q.front = ___0_____ q.rear = ____2_____

b) How many items can the queue contain at most?

Your answer : **____3_____**

3)(10 points) The following code describes some functions associated with the C implementation of the linked lists (LL). However, some parts of the code given below are missing. Fill in those missing parts indicated by _____ . The comments given before each function explains what that function is supposed to perform.

```
struct node
{
 int info;
 struct node *next;
};
int empty(struct node  ** plist)
{
 return * plist  = = NULL ? 1 : 0;
}
```

/* The following function inserts a node to the beginning of the linked list. */

```
void ins_begin(struct node  ** plist, int x)
{
 struct node  * p;
 p = (struct node *) malloc(sizeof(struct node));
 p  -> info = x;
 if(empty(plist))
 {
  *plist = p;  p -> next = NULL;
 }
 else
 {
  _____ p  -> next = *plist _____ ;

  *plist  = p;
 }
}
```

/* The following function returns a pointer to the first occurrence of x within the list and the NULL pointer if x does not occur in the list. */

```
struct node *search(struct node *list, int x)
{
 struct node *p;
 for(p = list; p != NULL;  ___ p = p -> next _____ )

  _ if(p -> info == x) _____ return p;

 return NULL;
}
```

/* The following function deletes the node after the node pointed to by p and stores the item in the deleted node at the location * px */

```
void del_after(struct node  * p, int  * px)
{
 struct node  * q;
 if (p = = NULL || p  -> next  = = NULL)
 {
  printf("void deletion\n");
  exit(1);
 }
 ___ q = p -> next _____ ;

 *px  = q  -> info;

 ___ p -> next = q -> next _____ ;

 free(q);
}
```

4)(7 points) The following code implements the removal of an item from a circular doubly linked list. However, some parts of the code are missing (indicated by _____). Complete the missing parts.

```
struct node
{
 int info;
 struct node *left, *right;
};

// The call delete(p, &x); deletes the node pointed to by p
// and stores its contents in x.
void delete(struct node *p, int *px)
{
  struct node *q, *r;
 if(p == NULL)
 {
 printf("void deletion\n");
 return:
 }
```

*px = ___ p -> info _____;

q = p -> left;

r = ___ p -> right _____;

___ q -> right _____ = r;

___ r -> left _____ = q;

free(p);
}

5)(10 points) Give the output of the following C program.

```
#include <stdio.h>
#include <stdlib.h>

struct node
{
  int info;
  struct node *next;
};

void fun1(struct node* list)
{
  if(list == NULL)
    return;
  printf("%d  ", list->info);

  if(list->next != NULL )
    fun1(list->next->next);
  printf("%d  ", list->info);
}

void add(struct node** list_ref, int new_info)
{
  struct node* new_node =
          (struct node*) malloc(sizeof(struct node));

  new_node->info  = new_info;

  new_node->next = *list_ref;

  *list_ref  = new_node;
}
```

```
main()
{
  struct node* list = NULL;

  add(&list, 5);
  add(&list, 4);
  add(&list, 3);
  add(&list, 2);
  add(&list, 1);

  fun1(list);
 }
```

1 3 5 5 3 1

6)(7 points) Assume that for the solution to a sorting problem with **n** items, you are given the algorithms **A, B, C**, and **D** such that those algorithms solve the problem after performing following number of operations:

A) $58n^2 + 100/n$ B) $n^3 + 32n + 40$

C) $118\ n\log_2 n$ D) $0.1(1.1)^n$

1.) Give the time efficiency (complexity) of the algorithms **A, B, C** in Big O notation (e.g. O(n)).

A) $O(n^2)$ B) $O(n^3)$ C) $O(n\log_2 n)$

2.) In case of large **n**:

i.) Which algorithm is the best one ? **C**

j.) Which algorithm is the second best one ? **A**

k.) Which algorithm is the third best one ? **B**