FIRST

94

جامعة بيرزيت

Computer Science Department

Comp232 First Hour Exam

Fall 1999

Iyad Jaber

Student Name: Dima Damen **Student ID:** 980037

- This question paper is composed of five pages containing four questions.
- Please read the whole paper before starting to answer.
- You have ten minutes to read the paper and then only twenty minutes to ask questions.
- The only questions allowed concern the spelling and the meaning of English words.
- Intermediate reasoning steps are demanded. Any unjustified answer is not accepted.
- Remember to write down you're your full name and your student id number.

Question1(25%)

Ordered lists are special lists that keep their elements sorted in an increasing order. An ordered list supports basically the following operations:

1. **Insert:**
   Inserts an element in the list at the right position so as to keep the list sorted in an increasing order.

2. **Delete:**
   Deletes a node from the list.

The structures and types to be used are given by the file OList.h:

```
/* Type declaration for linked list implementation of the OList.h */
#define FAILURE (0)
#define SUCCESS (1)

struct Node;
typedef struct Node *PtrNode;
typedef struct PtrNode OList;

struct Node {
        int Element;
        PtrNode Next;
}
```

Using linked list (with dummy header) implementation and the declarations given by OList.h, complete the two basic operations, Insert and Delete.

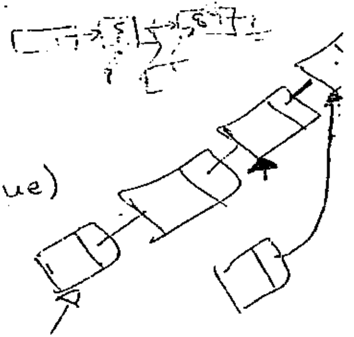
```
/* deletes the node pointed by the variable node from list*/
int Delete (OList list, PtrNode node)
{ PtrNode P; if (list==Null||node==Null) return failure
   P= FindPrevious (node,list);
   if (P->Next == NULL)
   { cout << "Node wasn't found";
       return FAILURE;
    }
    P->Next = node -> Next;
    free (node);
    return SUCCESS;
```

9/

2-7

1

```
/* Inserts a new element in the list
   while respecting the increasing order */

int Insert (OList list, int value)
{  PtrNode  p;  if (list == Null) return Failure;
   p = list;
   while (p->Next != NULL && p->Next->Element < value)
         p = p->Next ✓
   PtrNode  temp;
   temp = (PtrNode) malloc (sizeof (struct Node));
   if (temp == NULL)
        return FAILURE;
   temp->Next = p->Next;
   P->Next = temp;
   temp->Element = value;
   return SUCCESS;

}
```



```
PtrNode FindPrevious (PtrNode node, List list)
{  PtrNode P;
   P = list;
   while (P->Next != NULL && P->Next != node)
        P = P->Next;
   return P;
}
```

$n = \#$ of data $=$ hgh $-$ low $+ 1$

$$T(n) = \begin{cases} d, & n = 1 \\ c + T\left(\frac{n}{2}\right), & n > 1 \end{cases}$$

since the function divides the array in each move, into half

$$T\left(\frac{n}{2}\right) = c + T\left(\frac{n}{4}\right)$$

$$T(n) = 2c + T\left(\frac{n}{4}\right)$$

$$= ic + T\left(\frac{n}{2^i}\right)$$

$$\frac{n}{2^i} = 1 \Rightarrow i = \log_2 n$$

$$T(n) = \begin{cases} d, & n = 1 \\ T\left(\frac{n}{2}\right), & n > 0 \end{cases}$$

$$T\left(\frac{n}{2}\right) = T\left(\frac{n}{4}\right)$$

$$T(n) = T\left(\frac{n}{2^i}\right)$$

$$=$$

$$\frac{n}{2^i} = 1$$

$$n = 2^i$$

$$i = \log n$$

$$T(n_1, n_2) = \begin{cases} T\left(\frac{n_1+n_2}{2}+1, n_2\right) \\ T\left(n_1, \frac{n_1+n_2}{2}-1\right) \end{cases}$$

$$T\left(\frac{n_1+n_2}{2}, n_2\right) = T\left(\frac{n_1+2n_2}{2}, n_2\right)$$

$$= T\left(\frac{n_1+3n_2}{2}, n_2\right)$$

$$\frac{n_1+3n_2}{2} = 1 \qquad i = \frac{2-n_1}{n_2}$$

$$n_1 + in_2 = 2$$

Question 2(25%)

What is the worst case running time a function of n of the following code fragments.

a)

```
        for( j = 0; j < n; j++)                         n
                for( s = 0; s < n^2; s++)               n^2
                        for( k = j; k < j * j; k++)     (n^2-n)
                        {
                                t = j;
                                while (t > 0)
                                {
                                        for( d = 0; d < t; d++)      log n
                                                cout << d << endl;   log n * (n - 1)
                                        t /=2;
                                }
                        }
```

-3

? $n + \frac{n}{2} + \frac{n}{4} + \frac{n}{8} \dots < 2n$

$T(n) = n * n^2 * (n^2 - n) * \log n * (n-1)$

$= \Theta(n^6 \log n) = O(n^6 \log n)$

b)

```
int B_S ( int a[], int low, int high, int x)
{
C1          int mid;
C2          if ( low < high)
            {
C3                  mid = ( low + high ) / 2;
C4                  if (  a[mid] == x)
C5                          return mid;
C6                  else
C7                          if ( a[mid] < x)
C8                                  B_S ( a, mid+1, high, x);
C9                          else
C10                                 B_S( a, low, mid-1, x);

            }
C11         else
C12                 return -1;
}
```

since each traverse of the function divides the array into ha

assume n = # of data elements = high - low + 1

$$T(n) = \begin{cases} d, & n = 1 \\ T\left(\frac{n}{2}\right) + c, & n > 1 \end{cases}$$

$T\left(\frac{n}{2}\right) = T\left(\frac{n}{4}\right) + c$

$T(n) = T\left(\frac{n}{4}\right) + 2c$

$T(n) = iC + T\left(\frac{n}{2^i}\right)$

base case $\frac{n}{2^i} = 1$

$2^i = n \Rightarrow i = \log n$

$T(n) = c.\log n + T(1)$

$\Rightarrow = c.\log n + d$

$= O(\log n)$

text

A small text file to test my Algorithm

A

S

small

A

S

A small text file to test my algorithm

10

text

small

A

S

A small text file to test my algorithm EOF

file
text
small
A

S

to
file
text
small
A

S

test
to
file
text
small
A

S

my
test
to
file
text
small
A

S

algorithm
my
test
to
file
text
small
A

S

Second step

my
test
to
file
text
small
A

S

algorithm

output

algorithm my

output

test
to
file
text
small
A

S

algorithm my test

to
file
text
small
A

S

algorithm my test to

file
text
small
A

S

algorithm my test to file

text
small
A

S

algorithm my test to file text

small
A

S

algorithm my test to file texts

A

S

Last

algorithm my test to file text

~~to~~ small A

S

Question3(25%)

Write a pseudo-code function to read a sequential text file. Each line in the file contains a number of words. All words terminate with a white-space (' '). Print the words contained in the file in the reverse order in which you read them.

Suppose that you are provided with a function that returns the current word from the file and advances the file pointer to the next word. If EOF is reached, this function returns NULL.

```
char * ReadWord ()
```

1. You should very clearly describe the data structure that will be used to solve this problem and how the data structure should be applied (give the names of the basic operations supported by your data structure).

2. Apply the algorithm described in 1 on the following text file. Show clearly all the intermediate steps using figures to represent the consecutive states of your data structure.

A small text file to test my algorithm

Demo Text File

1. Data Structure is Stack, with Link List Implementation

Elementary Operations: used Stack because LIFO

1) push 2) pop ~~3) create sto~~ 3) Top 4) is_empty

```
void Print_reverse (ifstream ~~[illegible]~~ text, Stack S)
    char * String = ReadWord()
{   while (string != NULL)
    {  Push (S, String);
       String = ReadWord();
    }
    while ( ! is_empty (S))
    {    cout << Top (S) << ' ';
         Pop (S);
    }
    cout << endl;
    text.close();
}
```

14

Elementary Stack Operations:

*) Definition of Stack,

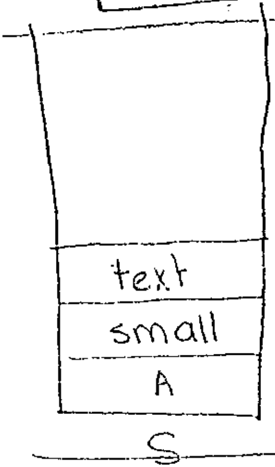
```
typedef struct node * nodeptr;
typedef nodeptr Stack;
typedef char * element_type;
```

```
struct node
{  ~~element_type~~ char * element;
   nodeptr next;
}

void push (stack S, ~~element_type~~ char *
{  nodeptr p;
   p = (nodeptr) malloc (sizeof (struct :
   if (P == NULL)
   {  cout << "Error";
      exit (1);
   }
   P -> element = x;
   P -> next = S -> next;
   S -> next = P;
}
```

24 +1

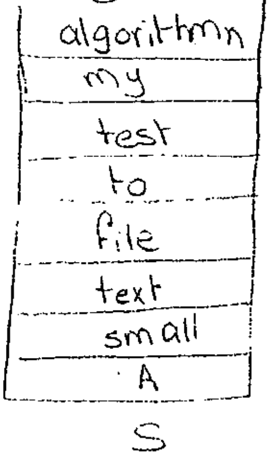
```
depth(p)
p = S->next;
  S->next = p->next;
  free(p);
}
```

```
  return(S->next == NULL);
}
element-type Top(stack S)
{ return S->next->element; }
```

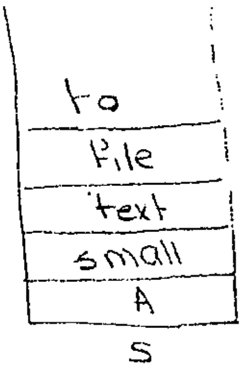


A small text file to test my algorithm

/* after three words are pushed into the stack */

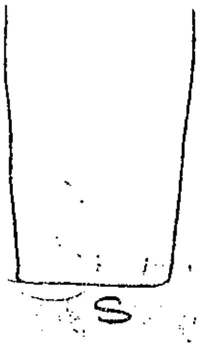


/* after all ~~the~~ file is read into the stack */



algorithm my test

output

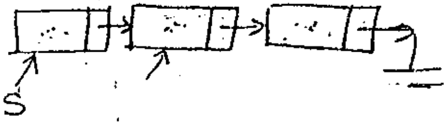


algorithm my test to file text small A

See at the back of previous page

Question 4(25%)

a- Write a recursive function, rec_display, to display the items in a stack on a single output line, with the bottom element printed first and the top element printed last. For efficiency, pass the stack by reference, but make sure the stack has been restored to its original state when the function terminates.



```
void rec_display (Stack S)
{     if (S -> next != NULL)
          rec_display (S -> next);
      cout << S -> element << ' ';

}
/* Note: this function receives the node next to the
   header not the header itself
   so in main ()
    {  stack S;
       rec_display (S -> next);
       cout << endl;
    }
*/
```

```
/* necessary declarations */
  typedef struct node * nodeptr;
   struct node
   {  element_type  element;
        nodeptr   next;
   }
   typedef  nodeptr  Stack;
```

$\frac{70}{100}$

Math. and Computer Department
Colmputer 232
First Hour Exam (Winter 1998)

Instructor: Nael Qaraeen

Student Name: [redacted] **Student id#:** [redacted]

(20%)

Q1) Give an analysis of the running time in terms of N for the following code fragments (Use *Big-Oh* notation)

a.

```
sum = 0;
for ( I=0; I < N; I++)          N
    for (j=0; j < I * I; j++)    N^1
        for (k=0; k<j; k++)  N^2
            sum++;
```

Order of magnitude = $O(N^5)$

b.

```
int  B_S(const Elementtype a[], Elementtype X, int N)
        {
            int low, mid, high;
            low = 0; high = N – 1;
            while (low <= high)
                {
                    mid = (low + high) /2
                    if (a[mid] < X)
                        low = mid + 1;
                    else
                    if (a[mid] > X)
                        high = mid – 1;
                    else
                        return mid;
                }
            return –1;
        }
```

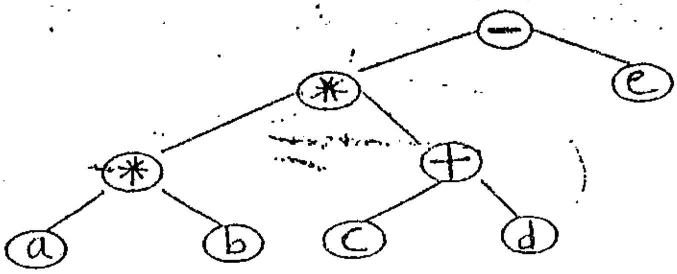
Order of magnitude = ~~[illegible]~~ O(Log N)

7-

(30%)

Q2)

a. Give the Infix and Postfix expressions corresponding to the tree below:



(I) Infix expression = a*b*(c+d)−e

(II) Postfix expression = ab* cd+* e −

b. Describe how you would insert an element at the bottom of a stack without affecting the order of the other elements already in the stack

~~we can't do that because the ordering the stack will affect~~

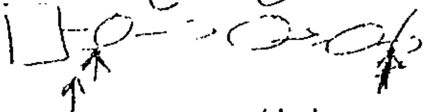
~~and we can do that by Putting all elements in the stack~~

~~and put the inserted the new element and by that we~~

~~affect the ordering the stack~~

we can do that by letting the stack to point to an array so we can make shift the element and put the new one in the bottom of the stack

linked list



8

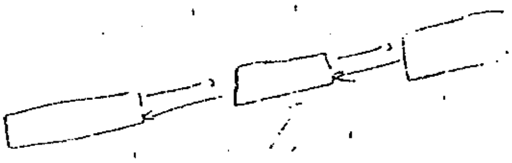
c. Suppose you have a queue ADT implemented as an array stored in k memory cells. When will this queue be empty in terms of the *Front* and *Rear* of the queue ?

it will ~~be~~ be empty when rear = front −1

(20%)

Q3) Given a ***doubly linked*** list of nodes. Using the following definitions, fill in the body of the insert function below which allows you to insert a node with element *X* ***after the node*** at position *P* in the doubly linked list *L*.

```
Struct Node;
typedef struct  Node *PtrToNode;
typedef struct  PtrToNode  List;
typedef struct  PtrToNode Position;

Struct Node
      {
              ElementType  element;
              Position   prev;
              Position   next;
      }
```



```
Position insert(ElementType X, List L, Position P)

{   Position    tmp, cell;
    Tmp = malloc((struct node));
     if (tmp==NULL)
       cout<<" error out of space);
     else
     {
       cell = P->next;
       tmp->element = x;
       tmp->next = cell;
       P->next = tmp;
       Tmp->prev= P;
       cell->prev= tmp;
     }
    return (P);

 }  */* End of function insert */*
```

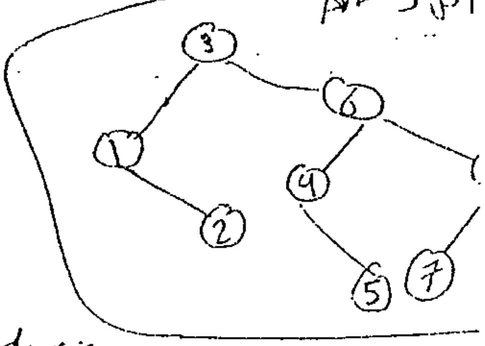
what if p->next

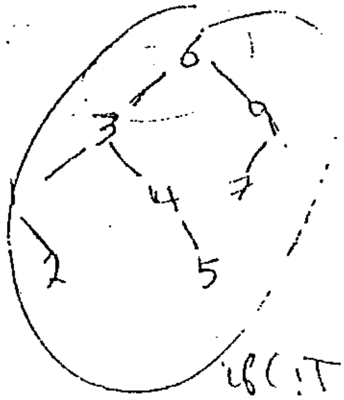
—4

—1

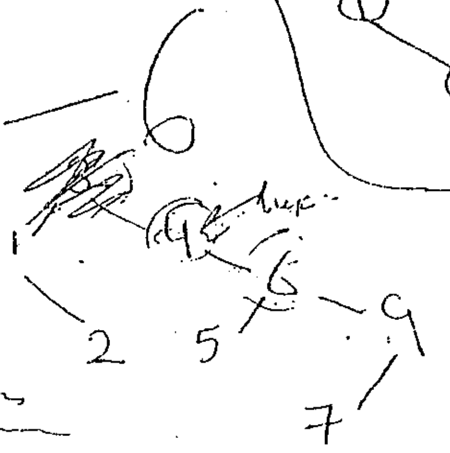
(30%)
Q4)

a. Show the result of inserting the following numbers *in order* into an initially empty binary search tree:
3, 1, 4, 6, 9, 2, 5, 7





if(!T)
if (X < T->elem)
T->left = del(



b. Is the tree generated in (a) above considered an AVL tree? Why? NO

~~Yes~~ Yes

نعم ، ذلك لانه من شروط AVL tree أن تكون Binary tree وأن يكون
الفرق بين الجزء اليمين واليسرى فقط edge واحد أو أن يكون Balance

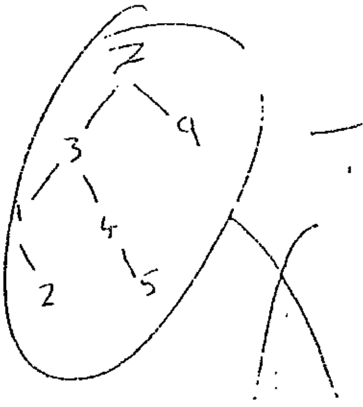
because the edges left side And right side of each Node must be such that (left - right) < 2 — 5

or 1 just

c. Show the result of deleting the root of the binary search tree in (a) above.

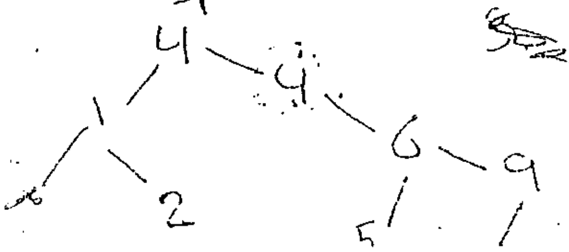
~~Here~~

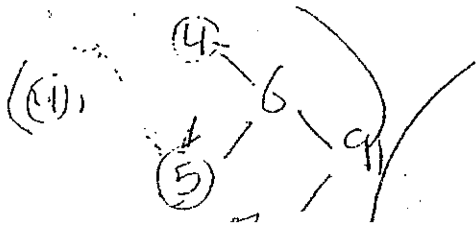
For each ~~sub tree~~ node In tree the hieght of left And right subtree must differ by at most 1



— 5

→ temp cell = 4





14

60

[redacted]

Computer Science Dept
Comp. 232
First Hour Exam

Instructor: Iyad Jaber   Date: 31/03/2000

Question #1(25%)

Suppose we have an array based list A[0..n-1] and we want to delete all duplicates. Last position is initially n-1, but gets smaller as elements are deleted. Consider the pseudo code program fragment in figure bellow. The procedure DELETE deletes the element in position J and collapses the list.

a- Explain how this procedure works.

b- Rewrite this procedure using general list operations.

c- Using a standard array implementation, what is the running time of this procedure?

d- What is the running time using a linked list implementation?

```
n    for ( int I=0; I< Last_Position; I++)
     {
1        int J= I+1;
n-1      while ( J < last_Position )
         {
1            if ( A[ I ] == A[ J ]);
                 Remove(J);
             else
                 J++;
         }
```

يعني كل عناصر ال Array الموجودة

يبدأ I [illegible] J

Time

$n(1+(n-1(1)))$

$= n(n$

$= n^2$

$T(n) = O(n^2)$

هذه loop [illegible]

كل عناصر A التي تلي I ويفحص ال A[I] تساوي

A[I] فاذا كانت تساوي سوف [illegible]

الثانية A[J]

اذا لم تكن تساوي فانه يبحث عن ما بعدها J++ ويقارنها [illegible]

ويعمل هذا loop [illegible] الى ان تصل آخر عنصر .

وبعد ذلك [illegible] I مع كل ما بعدها J+1 — J = last position

وما [illegible] I [illegible]

[illegible]

Best case [illegible]

$n^2 = n \times n$

~~position P next~~ P = A -> next

temp = A -> next

~~while (p -> next != Null && p -> element != temp -> element)~~

{

while (temp -> next != Null && temp -> element != p -> element

{ if (temp -> element == p -> element)

{ position pre

Pre = Find previous (A, temp -> element) ;

8

Question #2(25%)

What is the worst case running time a function of n of the following function:

$$T(n) = \begin{cases} 2T(n/2) + 10 & n > 1 \\ 1 & n = 1 \end{cases}$$

(Hint: $X^{I-1} + X^{I-2} + \ldots + 1 = (X^{I} - 1)/(X - 1)$

$$T(n) = 2T(\tfrac{n}{2}) + 10$$

$$= 2\left(2\left(T(\tfrac{n}{4})\right) + 10\right) + 10$$

$$= 4\left(2T(\tfrac{n}{8}) + 10\right) + 20 + 10$$

$$= 8\left(2\left(T(\tfrac{n}{16}) + 10\right)\right) + 40 + 20 + 10$$

$$\vdots \qquad 2^{K}\, T_{\frac{n}{2^{K}}} + \sum_{i=1} 2^{i-1}\, 10$$

$\frac{T}{2}(1)$ ستوقف عندما

$n = 2^{K} \Rightarrow K = \ln n$

$T(n) = n\, T(1) + 10 \sum_{i=1}^{\ln n} (2^{i-1})$ ⇐ K نعوض مكان كل

$= n + 10\,(\frac{2^{\ln n} - 1}{2 - 1})$

⇓

$T(n) = n + 10n - 1$

$= 11n - 1$

$T(n) > 11n$

$T(n) = O(n)$

~~$T_n = n + 10(\frac{2^{n} - 1}{2 - 1})$~~

~~$= n + 10(2^{n})$~~

25

① ~~a + 9/2 * 5 − b * 3 / 2~~

Question #3(25%) ②

a) Transform the following infix to postfix in details.

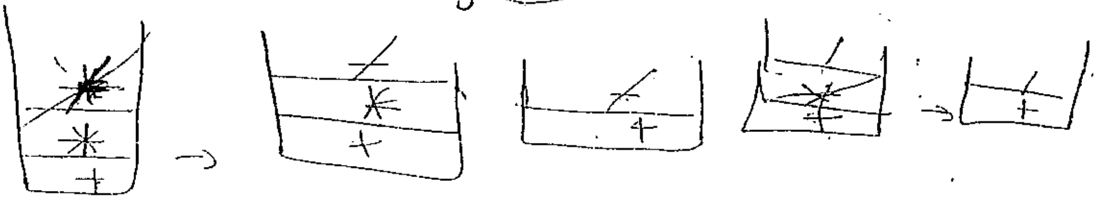
b) Transform the following postfix to infix in details.

a) a + 9/2 * 5 − b * 3 /2

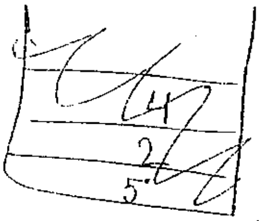
b) 5 2 4 / 3 * 9 / 2 / 7 * −

| a | 9 | 2 | ~~*~~ 5 * b ~~−~~ 3 * 2 / + |

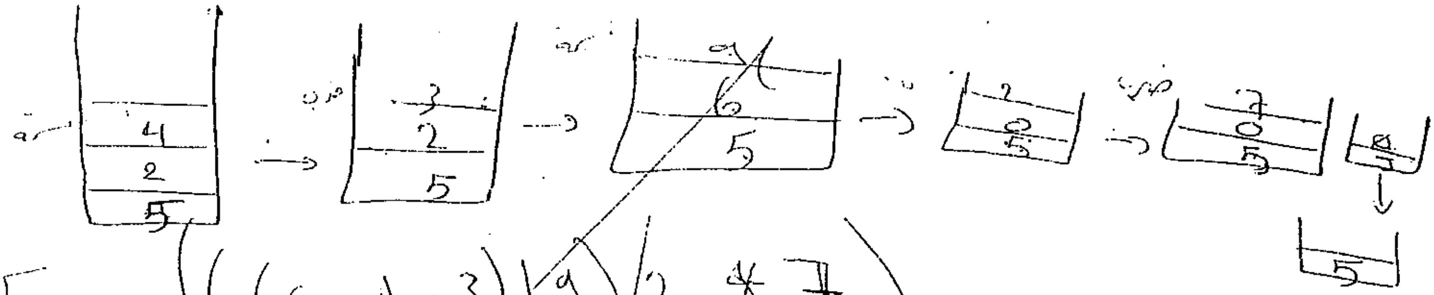
9 9 2 / 5 * + b 3 * 2 / −



stack



5 − ((2 / 4 * 3) / 9) / 2) * 7



5 − ((((2 / 4) * 3) / 9) / 2 * 7)

الجواب 5

18

Question #4(25%)

Write a routine addsame to add two integers of the same sign represented by doubly linked lists.

```
Typedef struct nod  * nock-ptr ;
Typedef int  element-type ;
struct nod
 {
    ~~pnode-ptr~~      ~~pre~~
    ~~nodeptr~~ .      ~~next~~.
    ~~element-type~~   ~~element~~
  } ;
 ~~Typedef  pode-ptr~~   position, ..
 ~~Typedef   node-ptr~~   Header ;
```

```
~~addsam~~ ~~element-type  add sum ( Header H )~~
~~{~~
    ~~position  temp ;~~
    ~~element-type  sum = 0 ;~~
    ~~temp = H~~
    ~~while ( temp -> next != Null )~~
      ~~{~~
        ~~sum + = temp -> elent~~
        ~~temp = temp -> next~~
      ~~}~~
    ~~return  sum ;~~
  ~~}~~
```

```
void Insert( Position P, elemt-type e)

 { position ~~temp~~ newnod;
   newnode =(positi )malloc ( sizeof( struct nod)),
     new node -> next = Null ;
  new node -> element = e ;

     P -> next = newnode;

 }
```

0

81/100

Computer Science Dept.
Computer 232
First Hour Exam

Instructor: Iyad Jaber
Date: 14/04/2003
Student Name: Haleem Abeed **No.** 1011005

Question #1(15%):

What is the worst-case running time a function of **n** for the following code segment?

```
for ( i = 0; i < n; i++)                (n)       n
{ for ( j = 0; j< i*i ; j++)            (n² - n)  n²    (n³)
  {
      for ( k = j ; k < j*j ; k++)      (n⁴ - n²) n⁴    n⁷
          cout << k;
      for ( s = i; s < i*i ; s++)       (n² - n)  n²    n⁵
          cout << s;
  }
```

$n^3 [n^4 + n^2]$

$T(n) = (n)\,(n^2 - n)\,[(n^4 - n^2) + \cancel{(n^2 - n)}]$

$T(n) = n^3 - n^2 \; [\, n^4 - n^2]$

$T(n) = n^7 + n^4 - n^5 - n^6$

$T(n) = O(n^7)$

$1 + \; 1 + 2n \, [1 + 2n^2] \, [\, 2n^4 + n^4 + 1 \; 2n^4 + n^2]$

15

Question #2(30%):

a) Given the following definitions for a linked list of nodes:

```
typedef struct node * nodeptr;
struct node
{
        int element;
        nodeptr   next;
};
typedef  nodeptr List, position;
```

Write a recursive function called find(LIST L, int x) which recursively looks for the integer x in lined list L and returns a pointer to x if it is found in L. Be sure to include any error checking necessary.

b) What is the worst case running time (Big_Oh notation) for the find function above?

```
position find (LIST L , int x ) {

  position P;
  P= L->next ;
   if ( P && P->element = x )
        return P;
  else
   if (P && P->element != x)
    return find ( P->next, x)
   else
    { cout<<" the elemnt." <<X <<" not fined \n";
       exit (1);
     }

   }
```

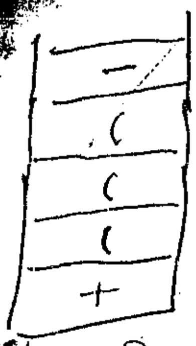
b//

27

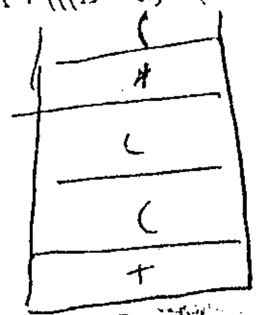
Question #3(25%):

Using a stack, transform the following infix expression to postfix. Show your work.

A + (((B – C) * (D – E + F) / G) * (H – J)) / K

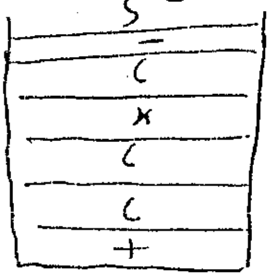


S ①

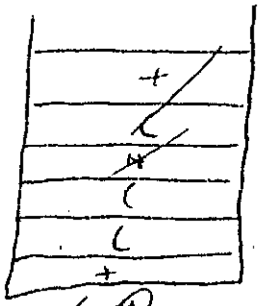


②

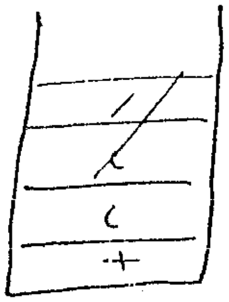
A B C – D E – F + * G / H J – * K



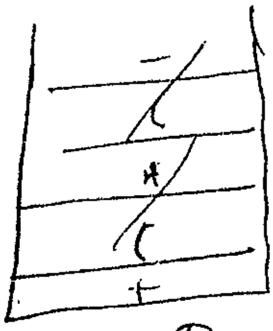
S ③



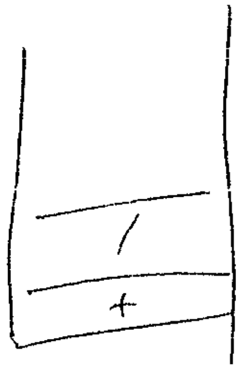
S ④



S ⑤



S ⑥



AB C – DE – F + * G / H J – * K ~~*~~ / +

2

Question #4(30%):

Given the following definitions for a linked list of nodes:

```
typedef struct node * nodeptr;
struct node
{
        int element;
        nodeptr   next;
};
typedef  nodeptr List, position;
```

b) time

Write a function called SortList (List L), which takes a pointer to the linked list L, and then sort the elements inside the list in ascending order.

```
void SortList( List L)
{
   posistion  P , temp ;     \\ temp is stack.
   P = L -> next ;     temp = (position)malloc (sizeof(node)) ;
   ~~while(P)~~          assert (temp) ;
   ~~{ temp -> next~~      temp -> next = NUll;

      while (P)            \\ n
      {   push ( temp , getsmal (P)) ;   \\ n
        cfree  (P , getsmal (P)) ;
      }

   L = temp ;
}.

int  getsmal ( Position p )
{   ~~while(P)~~  int y = p -> element ;

   while (P)
 {   if(y > p -> next -> element )
       swap (y , p -> next -> element );
     ~~else~~
      p = p -> next ;

   }  .
 return  y ;
```

$T(n) = (n)(n)$

$T(n) = O(n^2)$

```
void  swap (int &x, int &y)
{
 int  t = x ;
 x = y ;
 y = t ;
}
```

14

19/60

Computer Science Dept.
Computer 232

Instructor: Iyad Jaber **Date : 11/11/2003**

Student Name: Dalia Obeid **No.** 1011039

Question #1(20%):

Write a recursive function to add the first n terms of the series
+ 1/3! + 1/5! + 1/7! + ...

```
~~int~~ float add (~~int~~ n, sum)
                  1o
for ~~int add (int n, int sum~~
    ( i=0, i<n, i++) {
int Fact ( int ~~n~~, int ~~N~~)
 N=
    if (N == 0)
    return Sum=1;
    }
    els
    if (N > 0)
    return
 sum = { ~~n fact (n-1);~~ }
{s~~um~~{ n fact (n-1); }

   ~~Sum~~
   ~~sum = fact~~
 return { ~~Fact~~ ~~to~~
       ~~Sum = fact~~ n.

add (sum=) ~~sum~~ ;
 }
```

$1/2^n-1$

fact

$\begin{cases} 1 & N = 0 \\ n\,Fact(n-1) & N > 0 \end{cases}$

$\left(\frac{1}{2^n-1}\right)!$

n x

$\frac{1}{2^n-1}$ fact ($\frac{1}{2^{n-1}} - \frac{1}{2^{n-2}}$)$/1$

2^0

$N = \frac{1}{2^n-1}$

$\frac{1}{2^n-1}$ ~~K~~ $\left(\frac{1}{2^n-1}\right)! + \left(\frac{1}{2^n-1}\right)!$

~~[illegible]~~ g(o)

```
float add (int n, sum)
  for ( i=0, i<n, i++)
  add = fact n + ~~Fact~~ n
```

$2*(2)+1 = 5$

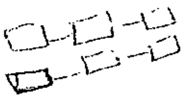
$2*(1)+1 = 3$

$2*(0)+1 = 1$

```
add (int n)
{
   if(n==0)
   return 0
 else
 return 2* ~~add (n-1)~~ ( 1/ add(n-1) ) +1;
```

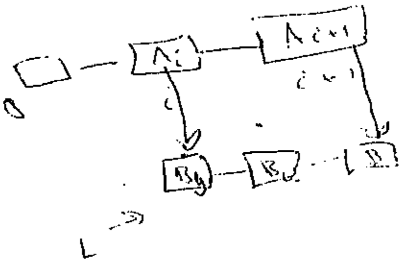
5

P = 1 3 4 6



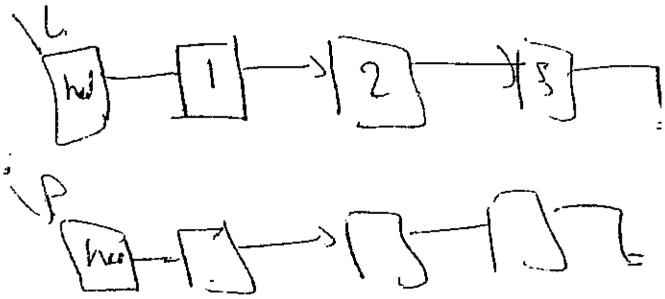
Question #2(20%):

You are given a linked list, L, and another linked list, P, containing integers, sorted in ascending order. The operation prin_lots(L,P) will print the elements in L that are in positions specified by P. for instance, if P = 1,3,4,6, the first, third, fourth, and sixth elements in L are printed. Write the routine print_lots(L,P). You should use only the basic list operations. **What is the running time of your routine?** O( )



```
Void Print_lots (L, P)
 position P, temp;
 ~~if~~ (
~~int~~  index n2,n1, i, j ;
     data * A, * B ;
     node * current ;
 if (P == Null)
    Cerr << error ;
    exit (1) ;
 if (P -> next == Null)
    return P ;
Current = malloc (sizeof c struct node) ;
  A[i] = 0
  i = 0
Current = current -> next
  i ++ ;

~~while (~~
For ( i = 0; i < n; i ++)
~~A[i]~~
j = A[i]
~~j++~~ ;
return B[j] ;
}
```

T(n) = O(n logn)



```
/n Void Print-lots(list L, list P
  {  ~~Position~~ ;
     if(P == 1)
  cout<<L->Next->elem ;
```

6

```
int nofNode( Tree T )
  int count =0;
  if(T==null)
    return 0;
  if(T->left==null&&T->right==null)
     return 1
  while(T->left!=null) // for left side)
  { count++;
     T=T->left;
     if(T->right!=null)
       coun++
  }
  while(T->right!=null) // for right side..
  {    count++;
       T=T->right;
       if(T->left!=null)
         coun++;
  }
  return count;
}
int NoL (Tree T)
{ if(T==0)
    return 0;
  while(T->left != null && T->right!=null)
  {  T=T-left;
     if(T->left && T->right==null)
       count++
  }}
```

Question #3(20%):

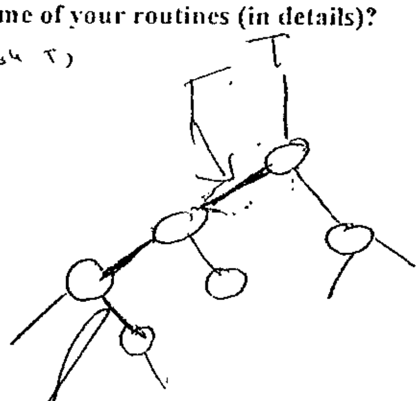
Write efficient functions that take only a pointer to a binary tree, T, and compute

a. The number of nodes in T.

b. The number of leaves in T.

What is the running time of your routines (in details)?

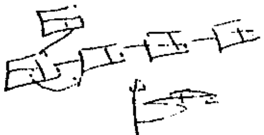
```
int node_Counter ( ~~search~~ T)
 treeptr = ~~T~~
if (T == Null)
 return T;
 ~~if (T~~.
if (child == 0)
 retur T;
als
if ( T→left && T→right)
 return ~~Count (T→left) + Count (T→right)~~
  Count T = count (T→left) + count (T→right)
els
if (T→left)
 retur
count T = count (T→left)
else
 if (T→right)
  return
   count T = Count (T→right)
 }
```



Coun
+, ~~X~~ ~~X~~ 1

• another
T→left = = Null
T→right = = Null

if T = = Null



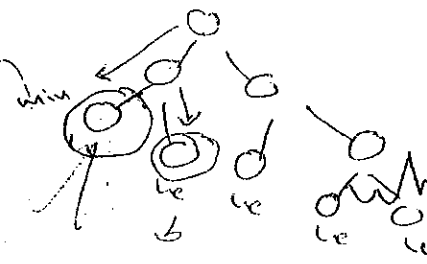
{ 0 , T =
 count , T .

if child

~~O (logn)~~ O(n) Because it will pass all the value while count them.

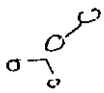
```
void number (Tree T)
{ int Cn, CL;
  if(T == Null)
  { Cn=0;
    CL=0;}
  else if(T→right != Null && T→left
    { Cn = 0
      CL = 0;}
  else if(T→right! = Null && T→le
```

```
int leavesCount ( leave )
 tree ptr. T;
if (T == Null)
 cerr << error
   exit(1);
if ( T→next == Null)
  return T;   // no leave
if( T→left != Null)
return
 leavesCount ;
Count = + count ;
if (T→ ~~le~~ righ == Null)
  return
  leaverCount ;
  Counter = + count ;
   }
```



```
if(T→left ==
{ Cn++;
  T = T→left
while(T→left
```

T → next == Nul



O (logn)

6

15/100

Computer Science Dept.

Comp. 232

First Hour Exam

Instructor: Iyad Jaber   Fahadeen Abed Almahdi 1020214   Date: 22/04/2004

Question #1(25%)

What is the worst case running time a function of n of the following function?

$$T(n) = \begin{bmatrix} 2T(n/2) + 10 & n > 1 \\ d & n = 1 \end{bmatrix}$$

max(x , L)

(Hint: $X^{l-1} + X^{l-2} + \ldots + 1 = (X^l - 1) / (X - 1)$)

$\frac{x^l - 1}{x - 1}$

$$T\left(\frac{n}{2}\right) = \begin{cases} 2T\left(\frac{n}{4}\right) + 10 & n > 1 \\ d & n = 1 \end{cases}$$

$T(n) =$ ~~$2(2T(\frac{n}{4})$~~ $2T\left(\frac{n}{2}\right) + 10 = 2\left(2T\left(\frac{n}{4}\right) + 10\right) + 10$

$= 4T\left(\frac{n}{4}\right) + 20 + 10$

$= 4T\left(\frac{n}{4}\right) + 30$

$= 2^2 T\left(\frac{n}{2^2}\right) + 2(1.5)$

$= 2^i * T\left(\frac{n}{2^i}\right) +$ ~~i~~

let $2^i = n \Rightarrow \log n = i$

2^i

~~$= nT(1) + 2\log n$~~

~~$= n(d) + 2\log n$~~

~~$= nd + 2\log n$~~

~~$O(n)$~~

8

Question #2(25%)

Write a recursive function called **divide(m, n)** which takes two positive integers, **m** and **n**. and returns the **n DIV m** without using "/" operation.

~~void divide (int m, int n)~~

{

divide { 0 $m < n$

($n > m$

~~divide (m, n) = { 0 m<n~~

~~div($\frac{m}{d}$, $\frac{n}{d}$) n>m~~

```
int   divide ( int m , int n)
{   If ( n < m)
      return 0
    else
      If ( n = = m)
        return 1;
    else
      If ( n > m )
        ~~return div( m/d , n/d )~~
        return ( divide( divide (m , d), divide (n, d)
    }
```

n DIV m

n Div m

$\frac{m}{n}$

$\frac{4}{2}$

$\frac{10}{5}$

$\frac{20}{10}$

(n+1)

$\frac{28}{7}$

m / n

n (3 n)

div(

div (m-1)

30% 10

$\frac{m-2}{n-3}$

$\frac{10}{5}$ = $\boxed{2}$

div ($\frac{m}{d}$, $\frac{m}{d}$)

$\frac{(10)}{(10)}$ = 1

div(n , n)

div($\frac{28}{7}$, $\frac{2}{28}$)

$\frac{10}{1}$ = $\frac{100}{10}$

n * d

$\frac{8}{(2)}$ = $\frac{2^2}{1}$

m * dive(m,

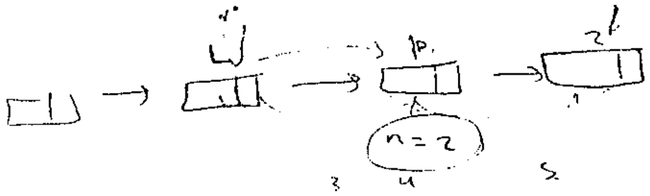
$\frac{(4)}{(2)}$ = $\frac{2^{(1)}}{0}$

div (mw(n, n-1), n))

Question #3(25%)

Write a function that receives a circular linked list **L** and an integer value **n**. The function should delete the **n**th element from the list starting from the **L** and then continues to delete the **n**th element circularly until the number of nodes becomes less than **n**.

```
~~delete (L~~

typedef struct node * ptr ;
struct node
{   elementtype   element ;
    ptr           next ;
};
typedef   ptr   list ;
typedef   ptr   position ;

void delete (list L , position , n) ;
{ while ( L != null)
      If (L -> next != null)
          L -> next = L -> next -> next ;
      If ( L -> next == P
```



```
void delete ( list L , int n)
{   position p ;
    p = L ;
    int s ;  // number of nodes النتيجة
    If ( s > n)
    {
        p = p + n ;
        free (p -> next) ; }
        s = s - n ; }
    else
    If ( n > s)
    cout << " go out " ;
}
```

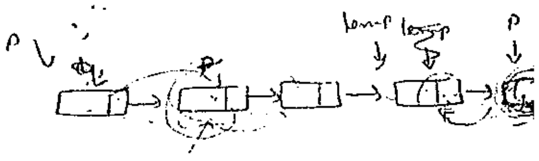
8

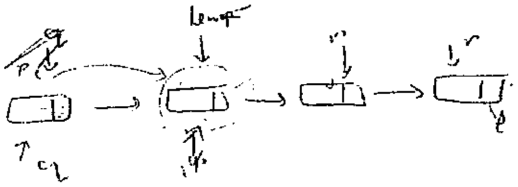
Question #4(25%)

Show how to implement a queue of integer in C++ by using linked list implementation, and then write the routines **remove, insert**, and **empty** for such an implementation.

```
typedef struct node *ptr,
struct node
{, element.type element;
  ptr ~~[illegible]~~ next ;
  };
 typedef ptr Queue ,
 typedef ptr position ;

void remove ( Queue q ) ~~elementtype~~
{, ~~q=~~ position p ,
      p = q ~~next~~ ,
      ~~If~~ ~~while~~ ( ~~p~~ ~~n~~ p -> next != null)
           p = p -> next
      else free(p); }
      ~~[illegible]~~
      ~~temp~~
  ~~temp = Find previous (P, x)~~
      ~~temp -> next =~~

void ~~elementtype~~ insert ( Queue q , elementtype x)
 {  position p
     p = q ,
     IF ( p -> next != null)
      p = p -> next ;
     else
   {  position temp ;
      temp (position) malloc (sizeof (struct node)) ;
       Assert (~~ptr~~ temp != null)

        ~~temp p ->~~
        ~~->~~          temp -> element = x ;
                        temp = p -> next ;
                        }

void  make.empty (Queue q)
{  position p , temp ;    IF (~~q next~~ p == null) ~~else~~ cout << "Queue is empty \n";
   ~~p = q -> next;~~                              else
   ~~temp~~                                       IF ( ~~p~~ ~~next~~ p != null)
   IF (~~q -> next != null~~)
```

17





$\frac{43}{105}$ FIRST

Computer Science Dept.
Computer 232

Instructor: Iyad Jaber **Date : 17/04/2007**
Student Name: [redacted] **No.** [redacted]

Question #1(25%):
What is the worst case running time a function of n of the following function?

```
void  confuse( int a[] , int left, int right)
{
        if ( left < right)
        {
                . . .
                . . .          }  T(n) = O(n)
                . . .
        }
        . . .
        Confuse( a, left , (left + right)/2 );   T(n/2)
        Confuse (a, (left + right)/2 , right );  T(n/2)
}
```

$$T(n) = \begin{cases} 1 & n = 1 \\ \cancel{2}T(\frac{n}{2}) + O(n) & n > 1 \end{cases}$$

$T(n) = \cancel{2}T(\frac{n}{2}) + n$

$T(\frac{n}{2}) = \cancel{2}T(\frac{n}{2^2}) + n \Longrightarrow$

~~$T(n) = 2(2T(\frac{n}{2^2}) + n) + n$~~

~~$T(n) = 2^2T(\frac{n}{2^2}) + 2n) + n$~~

~~$T(\frac{n}{4}) = T(\frac{n}{2^3}) + n$~~

$T(\frac{n}{4}) = 2(T(\frac{n}{2^3}) + n$

~~$T(n) = T(\frac{n}{2^3}) + 3n$~~

~~$T(n) = 2^2(2(\frac{n}{2^3}) + n) + 3n$~~

~~$= 2^3T(\frac{n}{2^3}) + 7n$~~

$T(n) = (T(\frac{n}{2^2}) + n) + n$

$T(\frac{n}{4}) = T(\frac{n}{2^3}) + n$

$T(n) = T(\frac{n}{2^3}) + n + 2n$

$\vdots$

$T(n) = T(\frac{n}{2^k}) + kn.$

let $n = 2^k \Rightarrow k = \log n$

$T(n) = T(1) + n \log_2 n$

$T(n) = 1 + n \log_2 n$

$\boxed{T(n) = O(n \log_2 n).}$

12

```
int check( char st[], int left, int right)
   { if (left<Right)
        if(st[left]!=[Right])
            return 0;
       else
         return (check(st,left+1,right+1);
         return1;
           }
```

Question #2(25%):

Write a recursive function to determine whether or not a string consisting of all uppercase letters is a palindrome. Note that 'FABBAF' and 'CEDEC' are palindromes, whereas 'GAEG' is not.

```
#include <string.h>

int palindrome ( char string, ~~int A~~)
        char A[ ];
{   int C = strlen (string) ;
    if (c % 2 == 0)
    return 0;

    else
    ~~return~~ ~~while(string != '\0')~~

      ~~return (~~
    ~~for(int, string = Null, ...string)~~
     ~~if (*s ==~~
for (int i = 0; i < c , i++, c--)
     if( A[i] == A[c])
      ~~return 1;~~   return 1; }
```

int palinrom

Not recursive

```
# include <string.h>
int palindrome (char string)
{ ~~string~~
   if(!is upper(string)) && strlen(string)%2 == 0 )
   return 0;
  else
   string = strcmp (string, string/2)
  ~~return~~
  ~~compare~~
   return palindrome (string) ;
   else return 1; }
```

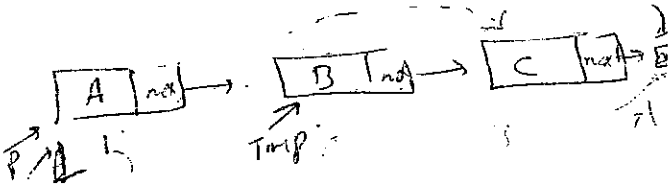
(8)

```
ptr reverse( ptr L)
 { ptr A, P, K ;
  A = L ;
   if ( L)

   { P -> L -> next ;

     A -> next = Null ;
     while( P)
     { K = P -> next
        P -> next = A ;
      A = P ;
      P = k ;
       }
     } return A ; }
```

Question #3(25%):

Write a procedure to reverse the pointers in a linked list. For example,
A→ B → C → D would become A← B ← C ← D. (don't use extra linked list)

```
type def struct Node ;
type def ~~No~~struct Node *ptrNode;
type def ptrNode list ;      type def ptrNode position;
struct Node{
 element_type  element;
 position   Next };
```



```
List reverse( list L)
 { position p , position tmp;

     p= L ~~next~~
 for( ; p!= Null;p=p → next) {
 ~~while(p!= Null) {~~

     tmp= p → next ;
   p → next = tmp → next
   Tmp → next = p → next → next ;
   if (tmp → next == Null){
   tmp → element = L → element
    L = L → next ; }

    return L; }
```

8

```
int hight(BST T)
{ if(T==Null)
     return-1;
   else
   return(max(hight(T→left)),hight(T→right))+1
   }

int leave(BT T)
 { if (T==Null)
      return 0;
    else if (T→left==Null && (T→Right==Null))
          return 1;
  else return (leaves(T→left)+leaves(T→Right));
         }
```

Question #4(25%):

Write efficient functions that take only a pointer to a binary tree, T, and compute

a. The height of the tree.
b. The number of leaves in T.

What is the running time of your routines (in details)?

```
~~Tr~~ ~~ty~~
type def struct tree ;
 "      "  tree *ptr;
 "      "  ptr  p;
struct tree {
~~Tr~~ element type element;
    P  left;
    p  Right;
   int hight}

int hight (Tree T)
         int counter
{  if(T == Null)
     return -1;
    else
if (~~T→Right~~ T→Right==Null & T → left == Null)
   return 0;
   else
   ~~if (T→Right~~
   return max(T → left, T→Right)
          }
```

```
int leaves (tree T)
        int
   { ~~while(T~~ counter;
   if(T == Null)
   cout << "no leaves";
 while~~if~~(T→Right // T→left)
   return 0 ~~counter++~~;
   };
```

$T(n) = O(n)$

لأنه لازم يمر على كل ال Nodes اللي
عندهم

```
int max(T→left , T→Right)
{ ~~while~~ ~~if~~(T→left != Null)
   hight ++;
    return Max(T→lef , T→Right);
else ~~if~~(T→Right != Null)
   high ++;
    retur Max(T→lef , T→Right)
    else return hight;
}
```

$T(n) = O(\log n)$

لأنه بدو يمر على أطول فرع
[illegible]
Binary
لا يعتمد على اليمين أو اليسار!

0