Computer Science Foundation Exam

Dec. 19, 2003

COMPUTER SCIENCE I

Section I A

No Calculators!

KEY

Name:____

SSN: ____

Score: / **50**

In this section of the exam, there are Three (3) problems

<u>You must do all of them.</u>

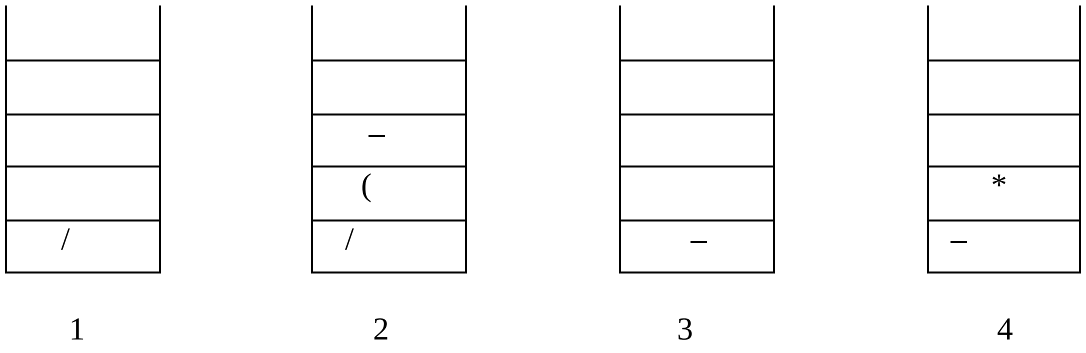
The weight of each problem in this section is indicated with the problem. The algorithms in this exam are written in C programming language notation. Partial credit cannot be given unless all work is shown.

As always, be complete, yet concise, and above all <u>be neat</u>. Credit cannot be given when your results are unreadable.

1. a) Convert the following infix expression to postfix, showing the values in the operator stack at each indicated point in the infix string (points 1, 2, 3 and 4), in the boxes below. You may add more boxes for each position if necessary to do so. **[10]**

A / B /^{1} (C – R – J ^{2} * M + N) – L ^{3} / D * E ^{4} * F

GRADING SCHEME:
Box 1 : **[1 point]**
Box 2, 3, 4 **[2 points each]**
Resultant expression **[3 points]**



Resulting Expression : A B / C R – J M * – N + / L D / E * F * –

b) A circular queue is implemented on an array of size N. Let tail represent the index of the last element added to the queue, and head represent the index of the element to be deleted from the queue. To start with , the queue is empty, and both tail and head equal – 1 .

How many more elements can the queue hold when **[10]**

i) head = 30 tail = 33 __N – 4 _________ **[3 points]**

ii) head = 3 tail = 3 ___N – 1 _________ **[3 points]**

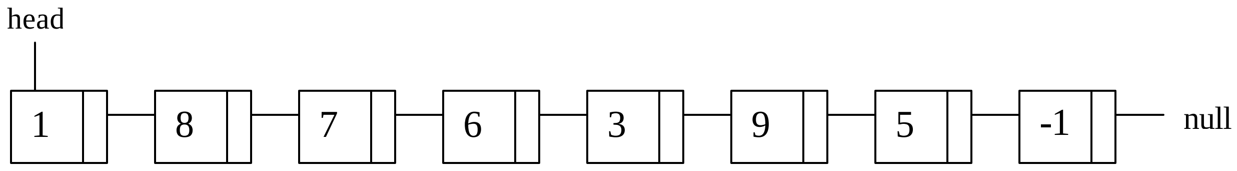
iii) head = 1 tail = N – 1 ____1________ **[2 points]**

iv) tail = – 1 head = – 1 ______N_______ **[2 points]**

2. Trace the effect of executing the code segment shown below on the linked list structure, also shown below, by drawing the linked list as it would look after the code has been executed **[10]**

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

p = head;
while (p != NULL)
{  if (p->data <= 6)
   {  newp = (struct list *) malloc(sizeof(struct list));
      newp->data = p->data + 8;
      newp->next = p->next;
      p->next = newp;
      p = newp;
   }
   else
      p = p->next;
}
```

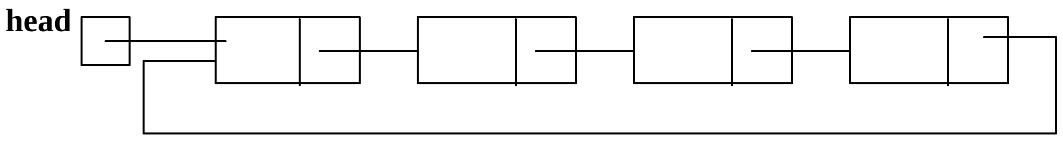


The new list has the elements

1, 9, 8, 7, 6, 14, 3, 11, 9, 5, 13, -1, 7

Give [2 points] for each new node created

3. Consider the following *circular list,* where head points to the FIRST node of the list. [20]



Assume the following declaration for nodes

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

Complete the following incomplete function to extend the circular linked list being pointed by p (the first node) by appending a singly linked list (pointed by q) to the end of the circular list.

<u>The singly linked list has got twice as many elements as the circular list.</u>

The resulting list will be one big circular list..

```
It is specified that

 * Circular list is NOT  empty i.e. p cannot be NULL,
   but q may be NULL.
```

- Desired: the first node of the original circular list will be the first node of the newly created big circular list again.

```
void extend(struct node *p, struct node * q)
{
   struct node *t; /* t is used to traverse the lists */

   if (   __ q == NULL   ____________ )  /* if standard
linked list is [ 2 points ]
                                   empty */
      return;
   t = p;      /* Assume p cannot be NULL */



  while (  t->next != p   )  /*traverse the circular list */
     [ 4 points ]


      t = __t ->next_______________;
   [ 2 points ]


  t->next = q_________; /* append the standard list */
   [ 2 points ]
   while (__t->next != NULL__) /* traverse the std. list */
     [ 2 points ]
      t = __t->next__________________;
     [ 2 points ]
   t->next = p_______________;    [ 2 points ]
/*here you may insert a statement to take care of the fact
that singly linked list contains twice as many nodes as
circular list. You may insert this statement anywhere else
too */
__________________________________


_________________________ [ 4 points if they leave this space blank ]______
}
```

Computer Science Foundation Exam

Dec. 19, 2003

COMPUTER SCIENCE I

Section I B

No Calculators!

KEY

Name: ____________________

SSN: ____________________

Score: ¤ **50**

In this section of the exam, there are four (4) problems

<u>You must do all of them.</u>

The weight of each problem in this section is indicated with the problem. The algorithms in this exam are written in C programming language notation. Partial credit cannot be given unless all work is shown.

As always, be complete, yet concise, and above all <u>be neat</u>. Credit cannot be given when your results are unreadable.

1a) An array contains a list of 1500 words arranged alphabetically. What is the minimum number of words that must be examined to establish that a particular word is not present in the list. **[3]**

Ans: Since it is alphabetically arranged, binary search can be applied which needs to examine the list $\log_2 n$ times, i.e. $\log_2 1500$.
Now $2^{10} = 1024$ and $2^{11} = 2048$
Since 1500 is more than 1024, it would require searching of 11 words.

1b) An array contains a sequence of n integers arranged in ascending order. Indicate a method to organize this data so that searching for an integer, deleting an integer or adding a new integer takes O(log n) operations. You may suggest a different data structure to hold the data, if you think it is necessary. In that case indicate how the data movement is to take place. **[10]**

Ans: The Binary search tree BST, is a data structure , where searching , deletion and insertion operations can be done in O(log n) operations. The data is in ascending order so if the elements are stored in that order in the BST tree, it will result in a skewed tree of height n. To get a balanced binary search tree, the root of the tree should be so chosen that the left half of the tree and the right half of the tree contain almost equal number of nodes. The subsequent nodes should also be chosen keeping this in mind.

<u>Data Movement :</u>

Let root be the integer at the middle of the array ,i.e. at the (n/2) th position. This will divide the array in two equal parts. The left child should be the middle element of the left part of the array , and the right child should be the middle element of the right part of the array. Keep on dividing the array at each stage and pick up the remaining elements in same manner.
Example: Let the array be 4 8 9 12 24 30 36 40 47 52 60 66 75 82 88 92 96

Root node: 47

```
                              47
  4 8  9 12  24 30  36  40          52  60  66 75 82 88 92 96
```

```
left child:  12                          right child : 75
                        47
             12                                   75
   4 8  9        24 30  36  40           52  60  66        82 88 92 96
```

The process continues with nodes at next level being 8, 30, 60 and 88
Final Binary search tree takes the following form:

```
                         47

             12                                   75

   8              30                     60                88

4      9      24      36              52      66       82      92

                          40                                        96
```

---

2. Carry out the following summations: **[15]**

$$\overset{40}{\underset{i = 1}{S}} \quad (2\, i - 4)$$

$= 2 [(40) (41)/2] - 4 [40]$
$= 40 (41 - 4)$
$= 1480$

$$\overset{14}{\underset{j = 3}{S}} \quad i^2 + 5 \ j$$

$= i^2 (14 - 3 + 1) + 5 [14(15)/2 - 2 (3)/2]$

$= 5 [105 - 3] +12 \ i^2$

$= 12 \ i^2 + 510$

$$\overset{n}{\underset{i = 1}{S}} \quad \overset{m}{\underset{k= 1}{S}} \quad \overset{k}{\underset{j = 1}{S}} \quad 22$$

$$\overset{n}{\underset{i = 1}{S}} \quad \overset{m}{\underset{k= 1}{S}} \quad 22 \ k$$

$$\overset{n}{\underset{i = 1}{S}} \quad 22 \ m(m+1)/2$$

$= 11 \ n \ m \ (m+1)$

---

3. Study the following algorithm for generating Fibonacci numbers, where the problem of generating the nth number is broken down into two parts, one trying to solve for (n – 1)th number and the other trying to solve for (n-2)th number. **[10]**

```
int fib( n )
{
  if ( n < 2 )
     return n ;
  else  return ( fib (n – 2 ) + fib ( n – 1 ) ;
}
```

What would be the complexity of this algorithm:

a) log n b) n c) n^2 d) n^3 e) more than n^3

Justify your answer.

Let T(n) be running time for calling the fib. Function.
For n =0 or 1 it takes constant time

$T(0) = T(1) = 1$

At any other stage, it involves calling fib function with running time T(n-1) and again with running time T(n-2). In addition, there is a "if" operation and one "addition" operation.
Thus,

$T(n) = T(n-1) + T(n-2) + 2$

Since fib (n) = fib (n-1) + fib (n-2)

it is easy to see that T(n) > fib(n), as at every stage T(n) is more than sum of T(n-1) and T(n-2).
It is known that fibonacci numbers grow very fast

$\text{fib}(n) > (3/2)^n$

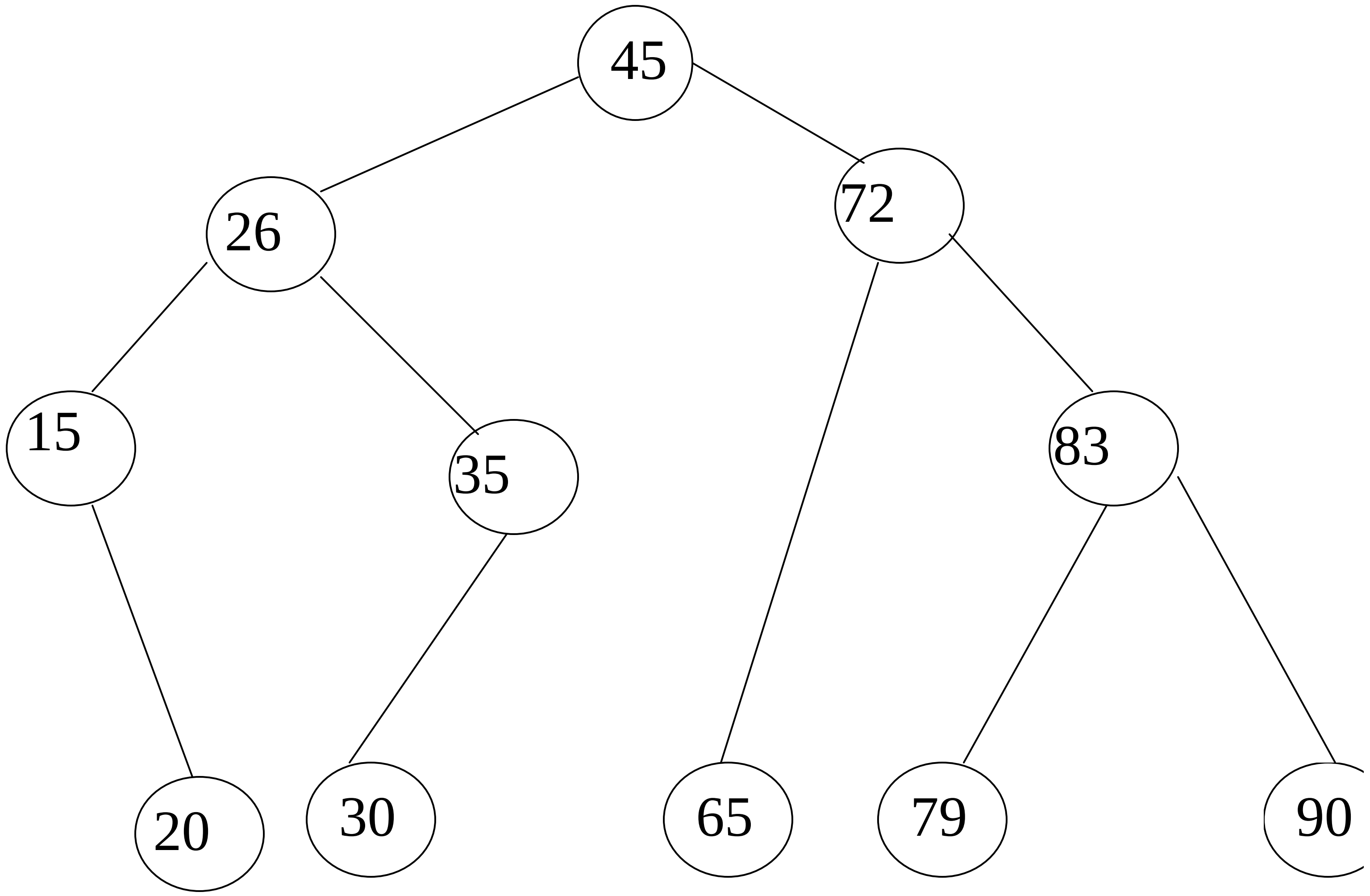
Thus,

$T(n) > (3/2)^n$
Which grows much faster than n^3 .

4. a)What will be the out put of the following algorithm , if it is used on the binary search tree shown in the figure . In the algorithm p points to the root node of the tree, and Q represents a queue.

```
If ( p ! = NULL) {
     Q.enqueue (p) ;
     while (Q is not empty) {
           p = Q.dequeue ;
           print ( contents of p ) ;
          if (  p -> left != NULL )
               Q.enqueue( p -> left) ;
                 if ( P -> right != NULL)
               Q.enqueue ( p ->right);
     }
}
```

4b) Write down the pre-order traversal of the tree **[12]**



Ans.

4a) 45, 26, 72, 15, 35, 65, 83, 20, 30, 79, 90

4b) 45, 26, 15, 20, 35, 30, 72, 65, 83, 79, 90