1) (10pts) Use Use the iteration technique to find the closed form solution of the following recurrence relation. After finding the closed form solution, give the solution in terms of Big-O notation.

$T(n) = 4T(n/4) + n \qquad T(1) = 4$

- $T(n/4) = 4T(n/4^2) + n/4$

2nd iteration $T(n) = 4[4T(n/4^2) + n/4] + n$

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

- $T(n/4^2) = 4T(n/4^3) + n/4^2$

3rd iteration $T(n) = 4^2[4T(n/4^3) + n/4^2] + 2n$

$= 4^3 T\left(\frac{n}{4^3}\right) + 3n$ ← start to see a pattern

k^{th} iteration $T(n) = 4^k T\left(\frac{n}{4^k}\right) + kn$

Since we know $T(1) = 4$ and we want to get rid of $T(\cdot)$'s from our expression, Let $n/4^k = 1 \rightarrow$

$n = 4^k \rightarrow \log_4 n = \log_4 4^k$

$k = \log_4 n$

$T(n) = 4^{\log_4 n} T(1) + n \log_4 n$

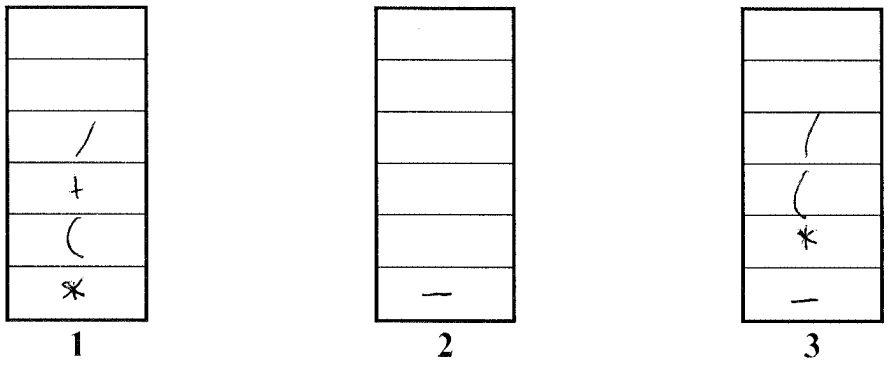
$= n^{\log_4 4} \cdot 4 + n \log_4 n = 4n + n \log_4 n$

Closed Form Solution: $4n + n \log_4 n$

Big-O Notation: $O(n \log n)$

2) (10 pts) Convert the following infix expression into its equivalent postfix expression using a stack. Show the contents of the operator stack at the indicated points in the infix expressions (points 1, 2 and 3), and also the final postfix expression. You may draw another stack alongside for your work.

A * (B + C / (D + E)) – F * (G / H + I)
1 2 3



postfix expression upto 1 : ABC

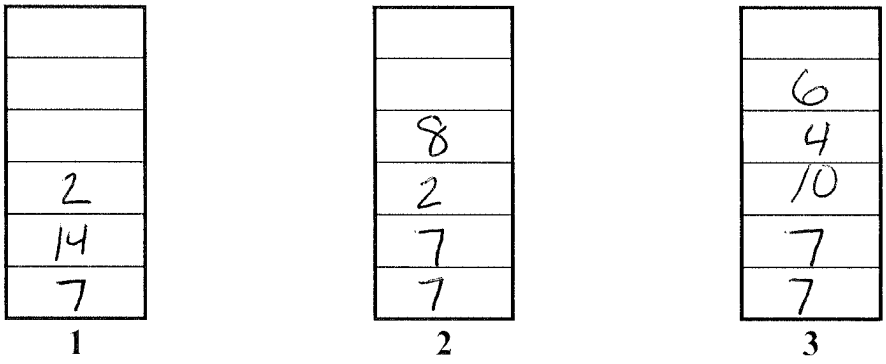
postfix expression upto 2 : ABCDE + / + *

postfix expression upto 3 : ABCDE + / + * F G

Final postfix expression : ABCDE + / + * FG H / I + * –

3) (8 pts) Stack Applications. Evaluate the following postfix expression using a stack. Additionally, you must show the contents of the stack at the indicated points (1, 2, and 3) in the postfix expression.

```
          1       2       3
7 14 2 / 2 8 + 4 6 5 + * + - -
```



54

4) (15 pts) Assume Q1 and Q2 are queues implemented using the array implementation discussed in class. Show the contents of Q1 and Q2 after the following code is executed, as well as the values for front and numElements.

```
typedef struct {
    int* elements;
    int front;
    int numElements;
    int queueSize;
} queue;

// Assume queueSize = 10 and elements is an array of size 10
// after the function call to init.
queue* Q1 = (queue*)malloc(sizeof(queue));
init(Q1);

queue* Q2 = (queue*)malloc(sizeof(queue));
init(Q2);

// Enqueue/Dequeue some items.
enqueue(Q1, 7);
enqueue(Q1, 8);
enqueue(Q1, 9);
enqueue(Q2, 1);
enqueue(Q2, 2);
enqueue(Q2, dequeue(Q1));
enqueue(Q2, dequeue(Q1));
enqueue(Q1, dequeue(Q2));
```

Q1: elements:

7	8	9	1						

front: 0 1 2

numElements: ~~3~~ 1 2

Q2: elements:

1	2	7	8						

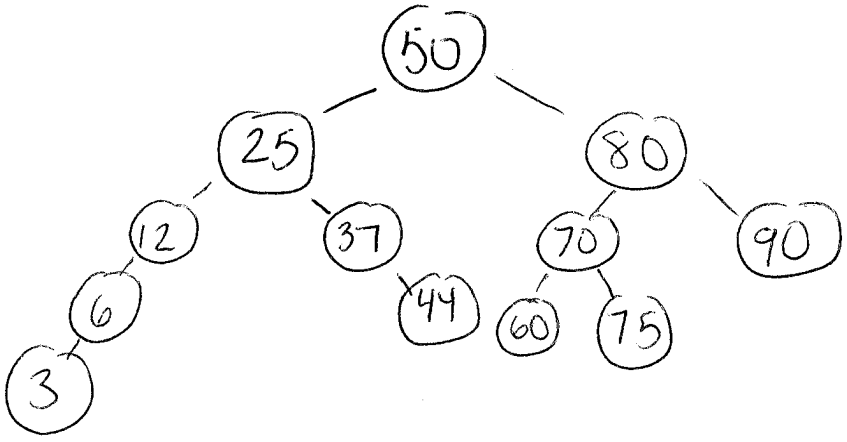
front: 0 1

numElements: ~~3~~ ~~4~~ 3

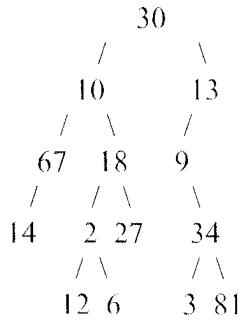
5) Using the operations on stacks (function prototypes given below), write a function that takes in an array of integers and then uses a stack to reverse the array and returns the new reversed array.

```
void initialize(stack* stackPtr);
int full(stack* stackPtr);
int push(stack* stackPtr, int value);
int empty(stack* stackPtr);
int pop(stack* stackPtr);
int top(stack* stackPtr);


int *reverseArray(int *array, int size) {

    Stack *myStack;
    initialize(myStack);
    int i=0;
    for(i=0; i< size; i++){
        Push(myStack, array[i]);
    }
    for(i=0; i< size; i++){
        array[i]= pop(myStack);
    }
    return array;

}
```

6) (5 pts) Draw a binary search tree created by inserting the following items in this order: 50, 25, 80, 70, 60, 75, 90, 12, 37, 6, 3 and 44.



7) (9 pts) Determine the preorder, inorder and postorder traversals of the following binary tree:

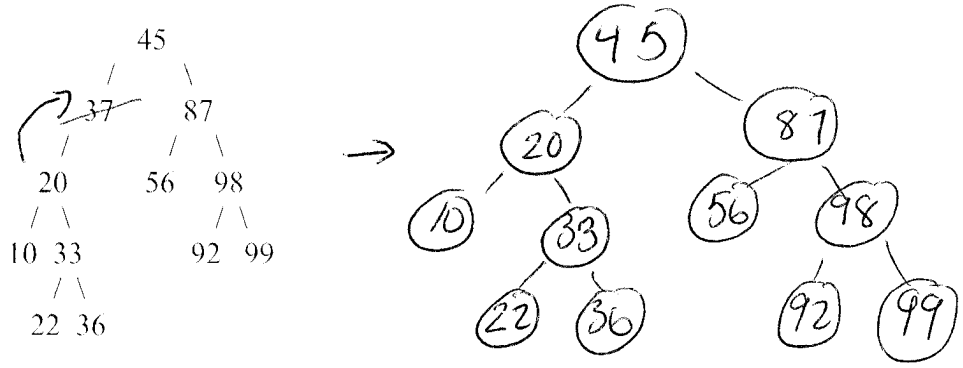


Preorder: 30, 10, 67, 14, 18, 2, 12, 6, 27, 13, 9, 34, 3, 81

Inorder: 14, 67, 10, 12, 2, 6, 18, 27, 30, 9, 3, 34, 81, 13, → shift down

Postorder: 14, 67, 12, 6, 2, 27, 18, 10, 3, 81, 34, 9, 13, 30

8) (5 pts) What is the result of deleting 37 from the binary tree depicted below?



9) Write a function that operates on a binary tree of integers. Your function should sum up the all of the even values in the leaf nodes only. Make use of the function prototype shown below.

Utilize the following struct that stores a binary search tree node for this question:

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

```
int sum_leaf_even(struct treeNode* p)
{
    if (p == NULL) return 0;
    if (p->data % 2 == 0 && p->left == NULL && p->right == NULL)
        return p->data;
    return sum_leaf_even(p->left) +
           sum_leaf_even(p->right);
}
```

10) (a)What is the result of converting 365_7 to base 9?

365_7 to base 10:

$365_7 = 3\times7^2 + 6\times7^1 + 5\times7^0$

$= 3\cdot 49 + 42 + 5 = 147 + 47 = \underline{194_{10}}$

194_{10} to base 9

$194 \% 9 = 5$

$21 \% 9 = 3$

$2 \% 9 = 2$

$9\overline{)194} = 21$ (18, 14, 9, 5)

$9\overline{)21} = 2$

$2/9 = 0$ STOP

$\boxed{235_9}$

(b)To what does 87 (in decimal) convert in binary?

87_{10} to binar

$87/2 = 43$

$43/2 = 21$

$21/2 = 10$

$10/2 = 5$

$5/2 = 2$

$2/2 = 1$

$1/2 = 0$ STOP

$87 \% 2 = 1$

$43 \% 2 = 1$

$21 \% 2 = 1$

$10 \% 2 = 0$

$5 \% 2 = 1$

$2 \% 2 = 0$

$1 \% 2 = 1$

$\boxed{1010111}$

$64 + 16 + 4 + 2 + 1 = 87$ ✓