Basic Algorithms – Homework 3 – due Sept 27/28

(1) Express the total time $T(n)$ for this program fragment as a sum and then evaluate the sum using Θ notation.

```
FOR I := 2 TO N
    J:= 1
    REPEAT J:= 2*J UNTIL J > I
```

(2) Find an $O(n)$ algorithm that takes as input a sorted array A with n elements and another number X, and determines if there are 2 elements in the array whose sum is exactly X.

(3) Solve the following recurrence relation exactly.

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

$$T(1) = 1$$

What is T(256)?

(4) Write a program to find a particular element in a singly linked list. Do this both recursively and nonrecursively, and compare running times. How big does the list have to be before the recursive version crashes?

(5) The function "subset" below takes two linked lists of integers and determines whether the first is a subset of the second. Give the asymptotic worst-case running time of subset as a function of the lengths of the two lists. When is this worst case achieved?

```
type numlist = record value : integer;
        next : ^numlist
end;

function element(X : integer; Q : ^numlist) : boolean;
         {Check whether integer X is an element of linked list Q}

var found : boolean; { Flag stating whether X has been found }
begin found := false;
```

```
         while (Q <> nil and not found) do
      begin found := Q^.value = X;
            Q := Q^.next;
      end;
         return(found)
   end;


   function subset(L,M : ^numlist) { Check whether L is a subset of M }

   var success : boolean; { Flag whether L is a subset so far }
   begin success := true;
         while (L <> nil) and success do
       begin  success := element(L^.value, M);
              L := L^.next;
end;
  return(success)
   end;
```