University of Toronto
CSC148F - Introduction to Computer Science – Fall 1996

FIRST TERM TEST

Borodin/Cook section

Duration: 50 minutes

No Aids allowed.

Make sure that your examination booklet has 5 pages (including this one).
Write your answers in the spaces provided. Write legibly.

Surname: _____ Given Names: _____

Student #: _____ Tutor: _____

1. _____/ 10
2. _____/ 23
3. _____/ 26

Total _____/ 59

[10] 1. Consider the problem of searching for an element in an array A. We have two programs. Program P assumes that the array is sorted and uses a binary search to determine if x is in A. Program Q does not assume that the array is sorted and uses a linear search. Below is a table of the running times for P and Q when they are run on an array A of size 1,000,000 and an element x that is not in A.

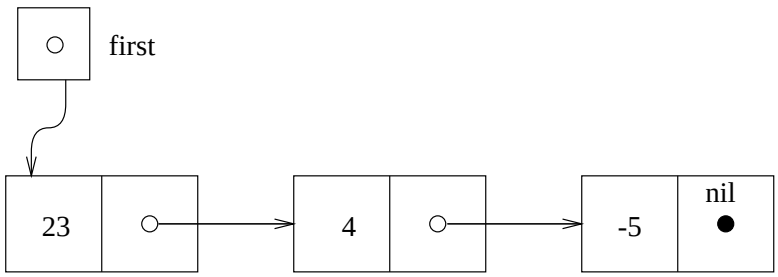
	Array A (1,000,000 elements)	Array B (4,000,000 elements)
Program P:	10 seconds	? (ANSWER : 11 seconds)
Program Q:	100 seconds	? (ANSWER : 400 seconds)

Now suppose P and Q are run on an array B of size 4,000,000 with an element that is not in B. Fill in the column B above for P and Q indicating for each one which of the following is the best estimate for the running time?

a) 5 seconds
b) 11 seconds
c) 20 seconds
d) 100 seconds
e) 200 seconds
f) 400 seconds

2. Assume that the type Box below has been declared, and that some OOT code has been executed to create the linked list shown below:

```
type Box:
   record
      value: int
      next : ^Box
   end record
var first: ^Box
```



[10] (a) Trace the following code and modify the diagram to show the state of the linked list and the variables front and temp after the following code has been executed.

```
var temp : ^Box
temp := first
temp -> value := 8
temp := temp -> next
temp -> value := 6
```

ANSWER:

"first" points to a node which contains the number 8 in its value field, and its next field points to a node whose value field contains the number 6 (which is also pointed to by "temp"). This node's next field points to another node whose value field contains -5 and whose next field contains nil. (Note: we would expect the student to draw the linked list rather than explain it in words.)

[5] (b) If the statement

```
put first -> value
```

is added at the end of the above code, what output would result?

ANSWER: 8

[8] (c) Add two more lines of code that delete the last node and free its memory locations. Make sure that what remains is a properly formed linked list.

ANSWER:

```
free (temp->next)
temp->next := nil
```

3. The BigInt module below is intended to perform operations on nonnegative integers with any number of decimal digits, subject to computer memory limitations. The implementation is a linked list of digits of the number, with the least significant digit at the front. (Picture the linked list going from right to left.) Leading zeros (which would be at the end of the linked list) are not stored. Each node stores a single digit. The number 0 is represented by the empty list (just the nil pointer).

Here is the signature of the module. (Procedure bodies are omitted.)

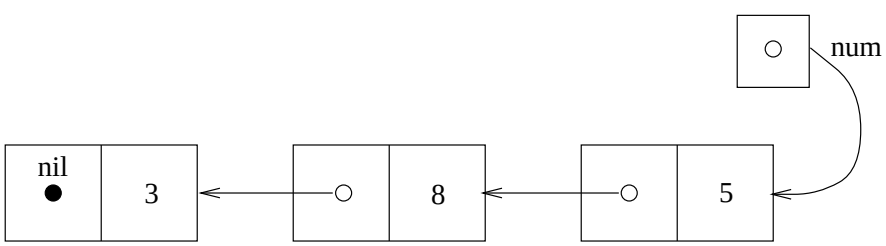
```
module BigInt

    export BigIntType, assignInt, add, timesTen

    type node :
        record
            digit : int
            link : ^node
        end record
```

```
    type BigIntType : ^node

    %Contructs a linked list num reprenting the integer newValue.
    %Precondition: newValue >= 0
    procedure assignInt (var num : BigIntType, newValue : int)

    % Returns num1 plus num2.
    function add (num1, num2 : BigIntType) : BigIntType

    % Returns 10 times num.
    function timesTen (num : BigIntType) : BigIntType

end BigInt

%Main Program goes here.
```

[10] (a) Suppose we want to define a function timesThree in the Main Program, using the add function from the module. Fill in the missing code below. (Two or three lines should be enough.)

```
%Returns three times num, using add from the module.
%Precondition:  num points to a list of digits in the proper format.

function timesThree (num : BigInt.BigIntType) : BigInt.BigIntType
    var : answer : BigInt.BigIntType
    % add code here:

         answer:=BigInt.Add(num,num)
         answer:=BigInt.Add(answer, num)


    result answer

end timesThree
```

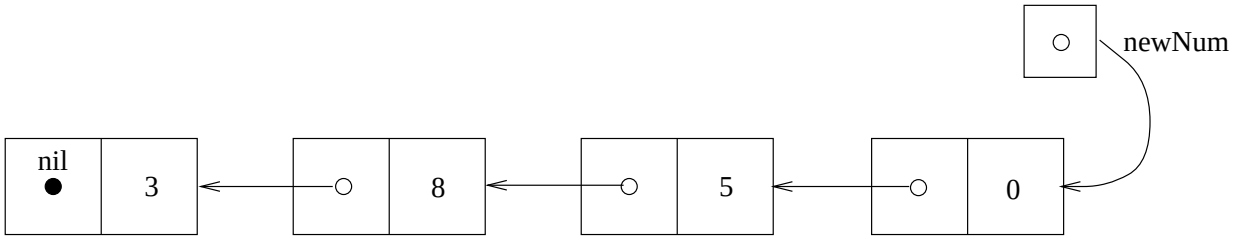
[16] (b) Complete the code on the next page for the module function timesTen by completing the four assignment statements indicated by *****. As an example, if some BigIntType variable num points to the list



then after executing

```
newNum := timesTen(num)
```

the original list for num is unchanged, and a new list for newNum is formed as follows:



```
% Returns 10 times num.
% Precondition: num points to a list in the proper format.

function timesTen (num : BigIntType) : BigIntType

    var answer: BigIntType

    % Ten times zero is zero
    if num = nil then
        answer := nil
    else
        % Create the digit zero as the least significant digit of answer.
        new answer
        answer -> digit := 0
        answer -> link := nil

        % Points to the current node in the original list
        var numpointer : BigIntType := num

        % Points to the last node in the new list
        var answerpointer : BigIntType := answer

        loop

           % create a new node of answer with the appropriate digit from num
           var temp : BigIntType
           new temp
*****      temp -> digit :=            ANSWER  numpointer->digit
*****      temp -> link :=             ANSWER  nil

           % Hook up the last node in answer to the new node and update
           % answerpointer
*****      answerpointer -> link :=    ANSWER  temp
*****      answerpointer :=            ANSWER  answerpointer->next

           % Move numpointer to the next digit.
           numpointer := numpointer -> link

           exit when numpointer = nil

        end loop
    end if

    result answer

end timesTen
```