CS 61b, Spring 94
Exam 1
Professor Clancy

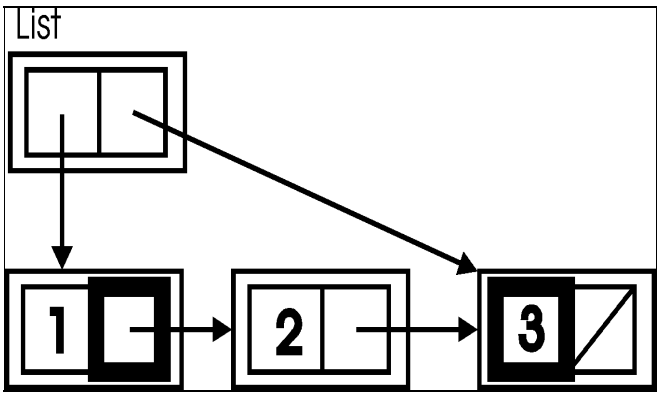
Problem 1 (2pts)

Consider the type definitions given below.

```
struct ListNode{
      int info;
      ListNode* next;
  };

  struct HeadNode{
      ListNode* first;
      ListNode* last;
  };
```

In the blanks below each storage location outlined in bold in the diagram, identify the C++ type of the storage location and, using the list variable, give an expression that names the storage location.



type: _____ type: _____
name: _____ name: _____

Problem 2 (8pts)

Recall that in lab assignments 2 and 3, you defined operations for the Date class:

```
enum Month { JANUARY, FEBRUARY, MARCH, APRIL,
  MAY, JUNE, JULY, AUGUST, SEPTEMBER,
  OCTOBER, NOVEMBER, DECEMBER};

  class Date {
  public:
```

```
        . . .   / / prototypes for public memeber functions go here
private:
    Month month;
    int day;
        . . .   / / prototypes for private memeber functions go here
};
```

In writing code for this problem, you may assume that member functions Equal, Increment, and Decrement have been defined for objects of class Date as in lab assignment 2-- Decrement will be similar to Increment, except that it subtracts 1 instead of adding 1--and that operators `++,--,==,<<` , and `int` have been defined for objects of class Date as in lab assignment 3.

Part a (5pts)

Provide a constructor function for class `Date` that, given an integer representing a day of the nonleap year (a value between 1 and 365, inclusive), constructs an object of type `Date` whose value is the date on the given day of the year. You may provide auxiliary functions as well.

All type conversions required in code that you write should be specified explicitly. (Thus g++ should not generate any warnings or error messages when compiling your code.)

Part b (3pts)

Redefine the `+=` operator so that when given an object date of class `Date` as first argument and an integer `numDays` as second argument, it replaces the value of date by the date `numDays` after date and returns the new date. For example, if date represents May 10, adding 31 to date should return June 10. Assume that the addition results in a date that's in the same calendar year as the given date.

Recall that in a member function, `*this` represents the value of the current object.

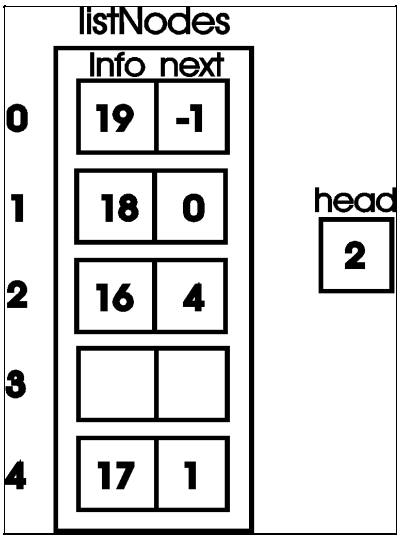
Problem 3 (4pts)

One can represent a linked list using an array of structs defined as follows:

```
const SIZE =  <some positive integer>;
       struct Node {
           int info;
           int next;
       };

       Node listNodes [SIZE];
       int head;
```

The links from node to node are represented by integer indices into the listNodes array, with a null link represented by the integer -1. The list (16 17 18 19), for instance, might be represented as follows:



Suppose now that the contents of `listNodes` represents a list of at least one element, the variable head contains the position of the node that represents the first element of the list, and the variable index contains the position of a node that represents an element to be inserted as the second element in the list. Provide, in the space below, a program segment that does the insertion.

Problem 4 (6pts)

Most of you used an auxiliary function in your vending machine prgram to decide if a given amount of change can be made from coins in the reservoir. Consider the following code, which might have constituted the body of such a function. (Assume that the Min function returns the smaller of its two argument values.)

```
// changeAmt contains the amount of change to return.
 // precondition: changeAmt >= 0.
 int nQ, nD, nN, temp;

 // Try to make change using as many quarters as possible.
 temp = changeAmt;
 nQ = Min (temp / 25, nQuarters);
 temp = temp - 25*nQ;
 nD = Min (temp / 10, nDimes);
 temp = temp - 10*nD;

 if (_________________________________) {
     return TRUE;

 }else if (_________________________________) {
     return FALSE;
 }

 // Try to make change using one fewer quarter.
 temp = changeAmt;
 nQ = Min (temp / 25, nQuarters) - 1;
 temp = temp - 25*nQ;
```

```
nD = Min (temp / 10, nDimes);
temp = temp - 10*nD;
return Boolean (temp / 5 <= n Nickels);
```

Part a (4pts)

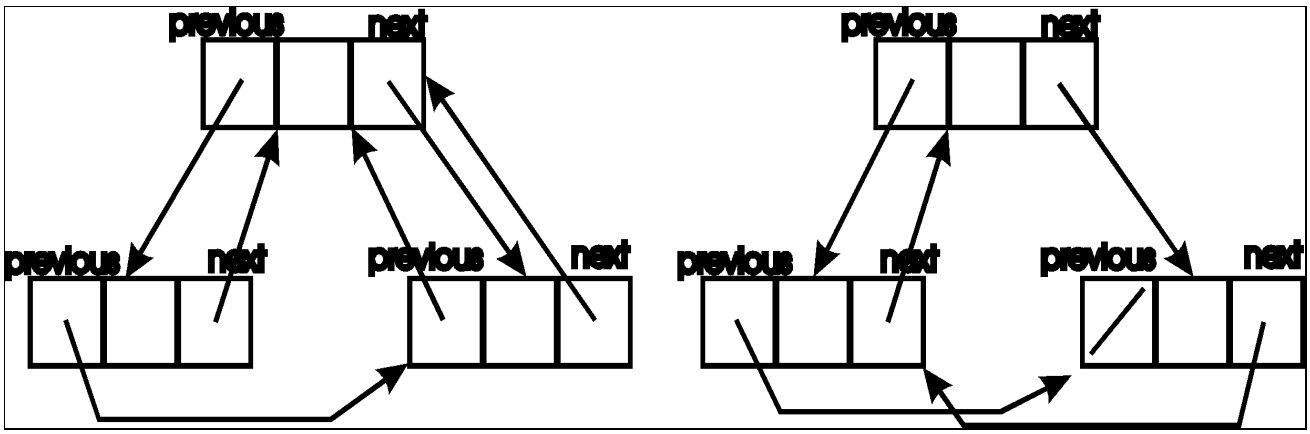
Fill in the blanks in the code above.

Part b (2pts)

In the "Check That Number" case study, a principle for simplifying code was described that the author of the code above attempted to follow. What principle is that and how was it applied in the code?

Problem 5 (8pts)

Write a function that, given a pointer to a node of type DLLnode, returns TRUE when that node is an element of a consistently linked circularly double-linked list, and returns FALSE otherwise. "Consistently linked means that the successor of the predecessor of each node is the node itself, and that the predecessor of the successor of each node is the node itself. Examples of inconsistently linked lists appear below, in each example, the pointer outlined in bold is inconsistent.



Assume that the following type definintions have been provided:

```
enum Boolean {FALSE=0, TRUE=1};

struct DLLnode {
        DLLnode* previous;        // points to node's predecessor
        int info;
        DLLnode* next;            // points to node's successor
```

Write your function on the next page. You may provide auxilliary functions as well.

```
//
//    Return TRUE when the node pointed to by p is
//    an element of a consistently linked cirular
//    double-linked list, and return FALSE otherwise.
//    Precondition : p != NULL
//

Boolean ConsistenlyLinked( DLLnode* p){
```