COMP 251 1996-97 Fall Semester Exam #2

Friday, Nov 15, 1996

Suggested Solutions

Important.

- **Please take a look at the problems now, and ask for clarifications if you don't understand them. After the first 30 minutes, no clarifications will be given.**
- **Spend your times wisely, and try to solve every problem because partial credits may be given for incomplete or even wrong answers.**

Problem 1. (10%) What value will the following Scheme expression return?

```
]=> '(a '(b))
```

Answer: `(a '(b))`

Problem 2. (15%) The following procedure, named `mystery`, takes as its argument a list that contains at least two top-level items:

```
(define mystery
  (lambda (ls)
     (if (null? (cddr ls))
         (cons (car ls) '())
         (cons (car ls) (mystery (cdr ls))))))
```

(a) (5%) What is the value of `(mystery '(1 2 3 4 5))`?

Answer: `(1 2 3 4)`

(b) (10%) Describe the general behavior of `mystery`, that is, for any list `(item1 item2 ... itemk)`, $k > 1$, what is the value of `(mystery '(item1 item2 ... itemk))`

Answer: `(item1 item2 ... item(k-1))`

Problem 3. (20%) Define a procedure `remove-last` that removes the last top-level occurrence of a given element `item` in a list `ls`. For example:

```
(remove-last 'a '()) ==> ()
(remove-last 'a '(b a n a)) ==> (b a n)
(remove-last 'a '(b a n (a))) ==> (b n (a))
```

Use `equal?` to test for sameness. Notice that you can only use the standard Scheme procedures in defining this procedure. But you can introduce any necessary helper functions as you wish.

Answer:

```
;;;member? function -- see lecture25 note
(define member?
 (lambda (item ls)
  (cond
    ((null? ls) #f)
    (else (or (equal? (car ls) item)
              (member? item (cdr ls)))))))
```

```
(define remove-last (lambda (item ls)
  (cond
    ((null? ls) '())                     ;base case
    ((and (equal? (car ls) item)         ;inductive case -- only when
          (not (member? item (cdr ls)))) ;item is (car ls) and does not appear
     (cdr ls))                           ;in (cdr ls) do we delete it from ls
    (else (cons (car ls)
                (remove-last item (cdr ls)))))))
```

The following are some Scheme constructs and standard procedures that you might need for solving the above problems.

```
'datum is a shorthand for (quote datum)
(cddr ls) is a shorthand for (cdr (cdr ls))
(equal? <obj1> <obj2>)
(if <cond> <expr1> <expr2>)
(define <var> <expr>)
(lambda (<var1> <var2> ... <vark>) <expr>)
(let ((<var1> <init1>) (<var2> <init2>) ... (<vark> <initk>)) <expr>)
(letrec ((<var1> <init1>) (<var2> <init2>) ... (<vark> <initk>)) <expr>)
(or <expr1> <expr2>) ;returns #f iff both <expr1> and <expr2> evaluate to #f
(and <expr1> <expr2>) ;returns #f iff either <expr1> or <expr2> evaluates to #f
(not <expr>) ;returns #t iff <expr> evaluates to #f
```

Problem 4. (15%) What are the problems with the following C++ program?

```
#include <iostream.h>
#include <string.h>
#include <assert.h>

class String {
 public:
  char *str;
  String(char*);
  ~String();
 private:
  int len;
}

String::String(char *s) {
  len = strlen(s); // strlen(s) returns the length of the string s
  str = new char[ len + 1 ];
  assert( str != 0 );
  strcpy( str, s );
}       <---- ``;'' MISSING

String::~String() { delete str; }

main() {
 String search( ``Rose'' );
   { String *p = &search;
     p->str = ``Rosebud'';          <--- STRING TOO LONG -- MAY OVERWRITE
                                         THE CONTENTS OF SOME LOCATIONS THAT
                                         HAVE BEEN USED FOR OTHER PURPOSES
     p->len = strlen(``Rosebud'');  <--- len: PRIVATE DATA MEMBER
     delete p;
   }
 cout << search.len << `` '' << search.str << endl;  <--- THE OBJECT THAT
                                                          WAS ASSIGNED TO
                                                          search HAS BEEN
                                                          DESTROYED BY PREVIOUS
                                                          delete STATEMENT
}
```

Problem 5. (40%) Consider the following hierarchy of numbers:

```
 number
   |
rational
   |
integer
```

Define a corresponding C++ class inheritance hierarchy: `Number` (base class), `Rational` (a class derived from `Number`), and `Integer` (a class derived from `Rational` and `Number`), and also a class `NumSeq` for holding sequences of numbers, so that the following main() program will run properly:

```
#include <iostream.h>
main() {
NumSeq x; // your default constructor for NumSeq should initialize x to
          // the empty sequence
Rational rat1(10, 11);  // declare a rational 10/11. This means that
          // your class Rational should have a constructor taking two
          // arguments of type int: the numerator and the denominator.
Integer i(20);  // declare an integer 20. This means that your class Integer
                // should have a constructor taking one argument of type int.
Rational rat2(10, 7);  // declare another rational 10/7.
x.print;  // this means that your class NumSeq should have a member function
          // called print. It prints out the numbers in the given sequence
          // separated by spaces. For empty sequence, which is the case for x here,
          // it should output ``Empty sequence.''
x.insert(rat1); // this means that your class NumSeq should have a member
              // function called insert. It insert a number to the front of
              // the sequence
x.print();      // there is only one number, the rational rat1, in x, so this
              // should print: 10/11
x.insert(rat2); // now x is the sequence (rat2 rat1)
x.print;       // so the output is: 10/7 10/11
x.insert(i);   // now x is the sequence (i rat2 rat1)
x.print();       // so the output is:  20 10/7 10/11
}
```

Answer:

```
#include <iostream.h>
#include <assert.h>

class Number {
 public:
  virtual print() {}
};
class Rational : public Number {
 int nu;
 int de;
 public:
  Rational(int x, int y) { nu = x; de = y; }
  print() { cout << nu << "/" << de; }
};
class Integer : public Rational {
 int value;
 public:
  Integer(int x) : Rational(x,1) {value = x;}
  print() { cout << value; }
};

class NumSeq {
 class Item {
  friend NumSeq;
  Number *num;
  Item *next;
 public:
  Item(Number &number) { num = &number; next = 0; }
};
 Item *numseq;
 public:
  NumSeq() { numseq = 0;}
  print() { Item *pt = numseq;
            if (pt == 0)  cout << "Empty sequence" << endl;
    else while (pt != 0) {
                (pt->num)->print();
                cout << " ";
                pt = pt->next;
              }
            cout << endl;
  }
  insert(Number &num) {
    Item *pt = new Item(num);
    assert( pt != 0);
    pt->next = numseq;
    numseq = pt;
  }
};
```

Warning. Cheating during an exam is a serious infringe on the University's policy on academic discipline, and can result in you being expelled from the University.