Burton Rosenberg

Midterm Answers

OUT: 19 OCTOBER 1993

1. Show that these two program fragments are identical. The variables `A` and `B` are declared as `boolean`, and `S1` and `S2` represent two statements.

```
Program Fragment 1:

  if A OR B then
    if A then S1
    else S2 ;

Program Fragment 2:

  if A then S1
    else if B then S2 ;
```

SOLUTION: Transform the first fragment to:

```
  if (A OR B) AND A then S1 ;
  if (A OR B) AND (NOT A) then S2;
```

The absorption identity gives:

$$(A \vee B) \wedge A = A,$$

which we can apply to the first `if`. Note: $\vee$ is the symbol for `OR`, $\wedge$ is the symbol for `AND`, and $\neg$ is the symbol for `NOT`. The law of distribution gives:

$$(A \vee B) \wedge (\neg A) = (A \wedge \neg A) \vee (B \wedge \neg A).$$

The first term of the OR on the right hand side is always false, so it reduces to only the second term. Therefore, we can transform our program again:

```
  if A then S1 ;
  if (B AND NOT A) then S2;
```

which is the same as:

```
if A then S1
  else if B then S2;
```

which is Program Fragment Two.

2. Change the following `repeat` loop into an exactly equivalent `while` loop.

```
{Precondition: N is any integer.}
i := 0 ;
repeat
  i := i + 1
until (i*i) > N ;
```

Solution: The formula is:

$$\text{repeat } \mathcal{S} \text{ until } \mathcal{C} \;\Leftrightarrow\; \mathcal{S};\ \text{while } \neg\mathcal{C} \text{ do } \mathcal{S}.$$

Applying the formula:

```
i := 0 ;
i := i + 1 ;
while not( (i*i)>N ) do
  i := i + 1 ;
```

We can neaten this up using simple identities:

```
i := 1 ;
while (i*i)<=N do
  i := i + 1 ;
```

3. Give code for the procedure

```
 Procedure Concat(A, B : List ) ;
```

which given two lists A and B, changes A into their concatenation and changes B into the empty list. Do this with as efficiently as possible.

```
Procedure Concat( a, b: list ) ;
begin
  if b^.first=nil then begin
    {there is nothing to do in this case}
  end else if a^.first=nil then begin
    {b is not empty, a is empty.}
    {copy b to a}
    a^.first := b^.first ;
    a^.last := b^.last ;
    {and make b empty}
    b^.first := nil ;
    b^.last := nil
  end else begin
    {both a and b are not empty}
    {connect the list together}
    a^.last^.next := b^.first ;
    {update list a}
    a^.last := b^.last ;
    {and make b empty}
    b^.first := nil ;
    b^.last := nil
  end
end;
```

But then we notice that the last three lines of the last two cases are identical, so we can pull them out and put them together:

```
Procedure Concat( a, b : List ) ;
begin
  if b^.first<>nil then begin
    if a^.first=nil then {a becomes b}
      a^.first := b^.first
    else {tack on non-empty b to non-empty a}
      a^.first^.next := b^.first ;
    {update a and make b nil}
    a^.last := b^.last ;
    b^.first := nil ;
    b^.last := nil
  end
end ;
```

4. Improve the speed in the inner loop of the following code fragment.

   (a) As written, how many multiplications are performed as a function of N.

   (b) Give an identically functioning code fragment where only $O(N)$ multipilications are performed.

```
var a : array[1..N,1..N] of integer ;
    i, j : integer ;
begin
  for i := 1 to N do
    for j := i to N do
      a[i,j] := i*i ;
end.
```

Solution: There are,

$$N + (N-1) + \ldots + 1 = (N+1)N/2,$$

multiplcations performed.

It would be best to pull the multiplication out of the inner loop, doing it one time for all just before the do loop:

```
for i := 1 to N do begin
  k := i*i ;
  for j := i to N do
    a[i,j] := k
end ;
```