CPSC 313
06W Term 2
Problem Set #2 Solution

1. The only illegal operation is taking the address of i, since it is in a register. Note the similarity of all of the accesses to the variables that are stored in memory: a, l, and g. Despite the differences in how their addresses are computed, the assembly code has the same pattern.

x	t = &x	t = x	t = *x	x = t	*x = t
i	illegal: i is in a register	movl %ecx, %esi	movl (%ecx), %esi	movl %esi, %ecx	movl %esi, (%ecx)
a	leal 8(%ebp), %esi	movl 8(%ebp), %esi	movl 8(%ebp), %ebx movl (%ebx), %esi	movl %esi, 8(%ebp)	movl 8(%ebp), %ebx movl %esi, (%ebx)
l	leal -4(%ebp), %esi	movl -4(%ebp), %esi	movl -4(%ebp), %ebx movl (%ebx), %esi	movl %esi, -4(%ebp)	movl -4(%ebp), %ebx movl %esi, (%ebx)
g	leal g, %esi	movl g, %esi	movl g, %ebx movl (%ebx), %esi	movl %esi, g	movl g, %ebx movl %esi, (%ebx)

2. (a) When $x < y$, it will compute first $x - y$ and then $y - x$. When $x >= y$, it just computes $x - y$.
   (b) The code for then-statement gets executed unconditionally. It then jumps over the code for else-statement if the test is false.
   (c)
```
    then-statement
    t = test-expr;
    if(t)
        goto done;
    else-statement
done:
```
   (d) The code in then-statement must not have any side effects, other than to set variables that are also set in else-statement.

3.
```
int switch_prob(int x)
{
    int result = x;

    switch(x) {
    case 50:
    case 52:
        result <<= 2;
        break;
    case 53:
        result >>= 2;
        break;
    case 54:
        result *= 3;
    /* Fall through */
    case 55:
        result *= result;
    /* Fall through */
    default:
        result += 10;
    }
    return result;
}
```

4. Max.s might look like this. I generated mine from a .c file.

```
        .file   "max.c"
        .text
        .p2align 4,,15
.globl max
        .type   max, @function
max:
        pushl   %ebp
        movl    %esp, %ebp
        pushl   %esi
        movl    8(%ebp), %esi
        pushl   %ebx
        movl    12(%ebp), %ebx
        movl    (%esi), %ecx
        cmpl    $1, %ebx
        jle     .L2
        movl    $1, %edx
        .p2align 4,,7
.L4:
        movl    (%esi,%edx,4), %eax
        cmpl    %eax, %ecx
        jge     .L5
        movl    %eax, %ecx
.L5:
        incl    %edx
        cmpl    %edx, %ebx
        jne     .L4
.L2:
        popl    %ebx
        movl    %ecx, %eax
        popl    %esi
        popl    %ebp
        ret
        .size   max, .-max
        .ident  "GCC: (GNU) 4.1.0 (SUSE Linux)"
        .section        .note.GNU-stack,"",@progbits
```

And a simple driver program that certainly doesn't extensively test max, but does at least call it, might look like this:

```
#define SIZE 100

extern int max(int a[], int size);

int main(int c, char **v)
{
    int a[SIZE];
    int i;
    for (i = 0; i < SIZE; i++) {
        if (i & 1)
            a[i] = i * i;
        else
            a[i] = (SIZE - i - 1) * (SIZE - i - 1);
    }
    int m = max(a, SIZE);
    printf ("Max is %d\n", m);
}
```