Nathan Tung
CS33 Dis 1C
April 25, 2012

CS33 Homework #3

3.59)
```
int switch_prob(int x, int n)
{
	int result = x;
	switch(n)
	{
		case 50:
		case 52:
		{
			result = result << 2;
			break;
		}
		case 53:
		{
			result = result >> 2;
			break;
		}
		case 54:
		{
			result = result + result*2;
			result = result*result;
			result = result + 10;
			break;
		}
		case 51:
		default:
		{
			result = result + 10;
			break;
		}
	}

return result;
}
```

In this rewritten C-code, I implemented a break into each of the cases. The machine code could also be translated literally to include the fall-through and achieve the same results.

3.64) A. The first value, 8(%ebp), contains the str2 result. The second and third value, 12(%ebp) and 16(%ebp), hold s1's integer a and pointer p (also called x and &y), respectively.

B. The first slot moves y pointer into %edx. The third slot moves the x value into %eax. The second slot holds the location of the initialized str1. The last bottommost two slots in the stack eventually hold the values of s1: that is, x and &y, respectively.

C. The structure values are stored next to each other in the registers but their memory positions are given to %esp (stack pointer) so that the values can be retrieved later according to its particular structure.

D. The function basically returns the structure's individual values but in accordance with the stack pointer in order to keep track of the structure and its ordering of values.

Nathan Tung
CS33 Dis 1C
April 25, 2012

3.67) A. Offset of…

e1.p: 0
e1.y: 4
e2.x: 0
e2.next: 4

B. In total, the structure requires 8 bytes (4 bytes for each part, then unioned.)

C.
```
void proc (union ele *up)
{
      up->e2.next->e1.p = *(up->e2.next->e1.p) - up->e1.p;
}
```

3.70) A.
```
long traverse(tree_ptr tp)
{
      long ans;
      if(tp==NULL)
            return 0x8000000000000000;
      long tpl = traverse(tp->left);
      long tpr = traverse(tp->right);
      if(x <= tp->val)
            ans = tp->val;
      else
            ans = x;

      if(y>ans)
            ans = y;
      return ans;
}
```

B. If a larger value is found in either side of the tree, update the variable ans. Continue to do this until the biggest value in the tree is found and returned. If the tree is empty, we return the maximum negative number (that is, TMin).