22C:050 HW#4 Solution

1. a) The Two-Pass Solution works for this forward referece problem, because the number of definition lines does not exceed the number of passes.

After the 1st pass:

```
EAL V0.0 by Douglas W. Jones; Mon Sep 29 22:25:36 2003

1             |
2             |. = #200
3 0200: ????  |  W A
                   ^
undefined
4             |C:
5             |. = A
                   ^
undefined
6 ????: ????  |  W B
                   ^
undefined
7             |. = C
8 0202: ????  |B:W A
                   ^
undefined
9             |A:
```

After the 2nd pass:

```
EAL V0.0 by Douglas W. Jones; Mon Sep 29 22:25:05 2003

1             |
2             |. = #200
3 0200: 0204  |  W A
4             |C:
5             |. = A
6 0204: 0202  |  W B
7             |. = C
8 0202: 0204  |B:W A
9             |A:
```

b) This source code needs passes of three times for B values to evenually for B value to fill in its position.

After the 1st pass:

```
EAL V0.0 by Douglas W. Jones; Mon Sep 29 22:20:06 2003

1             |
2             |. = #200
```

```
3 0200: ????  |  W B
                    ^
undefined
4             |C:
5             |. = A
                    ^
undefined
6 ????: ????  |B: W B
                     ^
undefined
7             |. = C
8 0202: ????  |  W A
                    ^
undefined
9             |A:
```

After the 2nd pass:

```
EAL V0.0 by Douglas W. Jones; Mon Sep 29 22:23:36 2003

1             |
2             |. = #200
3 0200: ????  |  W B
                    ^
undefined
4             |C:
5             |. = A
6 0204: 0204  |B: W B
7             |. = C
8 0202: 0204  |  W A
9             |A:
```

2. a) the displacement is -1D or 11111111B.
b)

```
...
5: begin { process 16 bit address instructions }
        M[location] := code;
        M[location] := displacement;
        location := location + 2;
   end;
```

3.

```
parse_expression()
{
        if(lex_this_val == ´(´)
        {
                parse_list();
        }
        else
        {
                parse_atom();
        }
}
```

```
parse_atom()
{
        lex_scan();
        if(lex_this.typ = IDENTIFIER || lex_this.typ = CONSTANT)
        {
                lext_scan();
        }
}

parse_list()
{
        lex_scan(); /* skip ´(´ */

        while(lex_this_val != ´)´);
        {
                parse_expression();

                if(lex_this.val != ´.´)
                {
                        lex_scan();
                        parse_expression();
                }

                lex_scan();
        }
        lex_scan(); /* skipp ´)´ */
}
```