Solutions to Homework #8

1. A DPDA M works as follows: before reading the first 1, M pushes three 0's to the stack for each 0 read. For each 1 after (and including) the first 1 read, M pops its stack. At this time, if a 0 is read, M aborts. At the end of the input word, M accepts if the stack is empty.

Here is the explicit construction:
$\Sigma = \{0, 1\}$, $\Gamma = \{0, Z_0\}$, $Q = \{q_0, q_1, q_2\}$, $A = \{q_2\}$.
Moves are:
$\delta(q_0, 0, *_1) = \{(q_0, 000*_1)\}$ ($*_1$ is 0 or Z_0)
$\delta(q_0, 1, 0) = \{(q_1, \Lambda)\}$
$\delta(q_1, 1, 0) = \{(q_1, \Lambda)\}$
$\delta(q_1, \Lambda, Z_0) = \{(q_2, Z_0)\}$
It is easy to check that this machine is deterministic.

2. A PDA M works as follows. M uses its stack as a counter; i.e., the stack has only one symbol a (besides the bottom symbol Z_0). The number of a's in the stack means the value of the counter. In order to maintain negative counter values, M' uses its finite control to keep track of the sign (+, -, or 0) of the counter. Initially, the sign is 0. When reading an input word x, if M reads a 0, it increments the counter by 1, if M reads a 1, it decrements the counter by 1. At the end of the input word, if the sign of the counter is + or 0, M accepts, else, M does not accept. An unsolved issue is how M can increment or decrement a counter by using the stack. Here is the trick. If the current sign is 0 or +, incrementing the counter is implemented by pushing an a to the stack and the sign changes to + if the current sign is 0. If the current sign is -, incrementing the counter is implemented by popping the stack. After the popping, if Z_0 is the top (i.e., the stack is empty), the sign changes to 0 from -. Decrementing the counter can be implemented similarly.

3. A PDA M works as follows. When reading an input word x, if M reads a 0, M pushes a 0 to the stack, if M reads a 1, and the current stack top is 0, M pops the stack. M accepts x if M reads the entire word x (i.e., M can read through the entire word without getting "stuck").
$Q = A = \{q_0\}$,
$\Sigma = \{0, 1\}, \Gamma = \{0, Z_0\}$,
$\delta(q_0, 0, *_1) = \{(q_0, 0*_1)\}, *_1 = Z_0, 0$,
$\delta(q_0, 1, 0) = \{(q_0, \Lambda)\}$

4. Remember M is a PDA accepting L. Now we describe how M' works in order to accept $Prefix(L)$. M', on reading an input word x, simulates M – this can be achieved by using the stack of M' simulating the stack of M. At the end of x, M' guesses input symbols for M and continue to simulate M on these guessed input symbols. At some moment, M' guesses that M enters an accepting state – then M' checks that the state of M is indeed an accepting state. If the checking result is 'yes', then M' accepts the input word x, else M' aborts (without accepting x). It can be seen that $x \in L(M')$ iff there is a y (this y is the guesses input symbols for M by M') such that $xy \in L(M)$. Therefore, $L(M') = Prefix(L)$.

5. Let Σ be the alphabet for type `char` in C. At least we know Σ contains a, b, c. L, the language accepted by the program, is $\{w \in \Sigma^* : \#_a(w) + \#_b(w) = \#_c(w)\}$. In order to construct a PDA M to accept L, the easiest way to is simulate the counter x by a stack, as we mentioned in the solution of Problem 5. Whenever you know how to do x++ and x– using the stack (look at Problem 5 solution), you already know how to translate the program into a PDA M.

Assume our stack has only one stack symbol '1' (besides the bottom Z_0). A counter is associated with the content of the stack (i.e., how many '1"s in the stack) and a sign (+,-,0). In the following, top() means the top symbol of the stack. x++, as we said before, is implemented by the following procedure (called plusplus()):

```
if  sign==+ then push('1');
else
    if  sign==- then
         {
         pop();
         if top()=Z_0 then sign:=0
         }
    else
        if  sign==0 then
            {
            push('1');
            sign:=+
            }
```

x– is implemented by the following procedure (called minusminus()):

```
if  sign==+ then
    {
    pop();
    if top()=Z_0 then sign:=0
    }
else
    if  sign==- then push('1');
    else
        if  sign==0 then
            {
            push('1');
            sign:=-
            }
```

Now the following code is indeed a PDA M that accepts L (I intentionally keep almost the same format of the C-code, but I replaced x++ and x– by the above stack implementations plusplus() and minusminus().):

```
sign:=0;

loop:
    read an input symbol d;
    if d is the end of input, then break the loop;
    if d is 'a', then call plusplus();
    if d is 'b', then call plusplus();
    if d is 'c', then call minusminus();
pool;


if top()=Z_0 then accept the input word;
else abort;
```