Solution to Homework 9

**Background.** In Problem 7, we considered a grammar with rules

\[ S \rightarrow \varepsilon \quad \text{and} \quad S \rightarrow cSd. \]

**Tasks:**

1. Use a general algorithm to construct a (non-deterministic) pushdown automaton that corresponds to context-free grammar described in Problem 7.

2. Show, step by step, how the word \( ccdd \) will be accepted by this automaton.

**Solution to Task 1.** By using the general algorithm, we get the following pushdown automaton:

![Pushdown Automaton Diagram](attachment:automaton.png)

**Solution to Task 2.** Let us show how this is done on the example of the word \( ccdd \) generated by the above automaton:

\[ S \rightarrow cSd \rightarrow ccSdd \rightarrow ccdd. \]

To make this derivation clearer, let us mark the variable \( S \) corresponding to different transitions by subscripts:
• we start with the first occurrence $S_1$ of the variable $S$;
• we then use the rule $S_1 \rightarrow cS_2d$ whose right-hand side contains the second occurrence $S_2$ of the variable $S$;
• this occurrence, in its turn, gets transformed into $S_2 \rightarrow cS_3d$ for yet another occurrence $S_3$ of the same variable $S$; so far, the derivation takes the form
  \[ S_1 \rightarrow cS_2d \rightarrow ccS_3dd; \]
• finally, to the occurrence $S_3$, we apply the rule $S_3 \rightarrow \varepsilon$, resulting in the desired derivation:
  \[ S_1 \rightarrow cS_2d \rightarrow ccS_3dd \rightarrow ccdd. \]

Let us now trace what our pushdown automaton will do. We start in the state $s$ with an empty stack:

The only thing we can do when in the state $s$ is push the dollar sign into the stack and get to the intermediate state $i$:  

![Diagram of the pushdown automaton]
The contents of the stack is as follows:

\[ \$ \]

When we are in the state \( i \), the only thing we can do is push the starting variable \( S \) (which corresponds to the first occurrence \( S_1 \) of this variable) into the stack and go into the working state \( w \):

\[ \varepsilon, \varepsilon \rightarrow S \]

Now, the stack contains the starting variable on top of the dollar sign:

\[ S \]

\[ \$ \]

Now that we are in the working state, we can start following the rules that were used to derive the word \( ccdd \). The first rule was \( S \rightarrow cSd \), or, to be precise, \( S_1 \rightarrow cS_2d \). As we have mentioned, this rule is implemented in three steps:
• first, we pop $S$ (that corresponds to the first occurrence $S_1$) and push the last symbol of the right-hand side – in this cases, the letter $d$ – into the stack, getting into the auxiliary state $a_1$;
• then, we push $S$ (that corresponds to the second occurrence $S_2$) into the stack, getting into the auxiliary state $a_2$;
• finally, we push $c$ into the stack, and go back to the working state $w$.

Let us illustrate this step by step.

First, we pop $S$, push $d$, and go into the state $a_1$:

The stack will now have $d$ instead of $S$:

Then, we push $S$ (corresponding to $S_2$) into the stack and go into the state $a_2$:

The stack will now have $S$ on top of its previous contents:
Finally, we push $c$ into the stack, and go back to the working state:

The stack will now have letter $c$ at the top:

Now, the letter $c$ is top of the stack. The only thing we can do if a terminal symbol is on top of the stack is use one of the rules of the type $x, x \rightarrow \varepsilon$ where $x$ stands for the corresponding terminal symbol.

In our case:
- since the terminal symbol on top of the stack is the letter $c$,
- we need to use the rule $c, c \rightarrow \varepsilon$,

i.e., we read the letter $c$ from the original word $ccdd$ and pop the top symbol $c$ from the stack:
After this popping, the variable $S$ (corresponding to $S_2$) will be on top of the stack:

$$
\begin{array}{c}
S \\
d \\
S
\end{array}
$$

According to the original derivation of the word $ccdd$, to get rid of the second occurrence $S_2$ of the variable $S$, we also use the rule $S \rightarrow cSd$, or, to be precise $S_2 \rightarrow cS_3d$. So, similarly to what we have before when we used this rule, first, we pop $S$, push $d$, and go to the state $a_1$:

Now, we have $d$ instead of $S$ on top of the stack:

$$
\begin{array}{c}
d \\
d \\
S
\end{array}
$$

After that, we push $S$ (that corresponds to the third occurrence $S_3$) and go to state $a_2$:
The stack now has the form:

```
S
d
d
$  
```

Finally, we push $c$ into the stack and go back to the working state $w$:

```
- ε, ε → S
- ε, ε → c
- ε, c → ε
- d, d → ε
```

Now, the stack has the following form:

```
c
S
d
d
$  
```

Now again, we have a terminal symbol $c$ on top of the stack, so the only thing we can do is use the rule $c, c → ε$: we read the second letter $c$ of the word $ccdd$ (the first one we have already read, so the cursor points to the second one) and pop $c$ from the stack. As a result, we get the following state:

```
- ε, ε → S
- ε, ε → d
```

```
- ε, ε → c
- ε, c → ε
- d, d → ε
```

```
- ε, S → d
- ε, S → ε
```

```
- ε, ε → c
- ε, c → ε
- d, d → ε
```

```
- ε, S → ε
```

```
- ε, S → ε
```

```
- ε, ε → c
- ε, c → ε
- d, d → ε
```

```
- ε, S → ε
```

```
- ε, ε → S
- ε, S → ε
```

```
- ε, ε → S
- ε, S → ε
```

```
- ε, ε → S
- ε, S → ε
```
The stack has the following form:

\[ S \]
\[ d \]
\[ d \]
\[ $ \]

On top of the stack is the variable \( S \) corresponding to the third occurrence \( S_3 \). To get rid of this variable, in the original derivation of the sequence, we used the rule \( S \rightarrow \varepsilon \) – or, to be more precise, \( S_3 \rightarrow \varepsilon \). This rule of the grammar corresponds to the transition \( \varepsilon, S \rightarrow \varepsilon \) of the pushdown automaton, i.e., we pop \( S \) from the stack:

The stack now takes the following form:

\[ d \]
\[ d \]
\[ $ \]

There is a terminal symbol on top of the stack – in this case, the symbol \( d \). We want an empty stack at the end. The only way to get rid of \( d \) is to use the rule \( d, d \rightarrow \varepsilon \), i.e., to read the next letter \( d \) from the word \( cccd \), and to pop \( d \) from the stack:
Now, the stack has the following form:

![Diagram]

Again, we have a terminal symbol $d$ on top of the stack, so we again use the rule $d, d \rightarrow \varepsilon$, i.e., we read the next letter $d$ of the word $ccdd$, and we pop $d$ from the stack:

![Diagram]

Now, the stack only contains the dollar sign:

![Diagram]

We have read all the letters of the original word, and all we have in the stack is the dollar sign. So now, we can use the rule $\varepsilon, \$ \rightarrow \varepsilon$ to pop the dollar sign and to go to the final state:

![Diagram]
Now, we are in the final state $f$ with the empty stack. This means that the word $ccdd$ is accepted by this pushdown automaton.

**A graphical description of the transitions.**

<table>
<thead>
<tr>
<th>read</th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>state</td>
<td>$s$</td>
<td>$i$</td>
<td>$w$</td>
<td>$a_1$</td>
<td>$a_2$</td>
<td>$w$</td>
<td>$-$</td>
<td>$w$</td>
</tr>
</tbody>
</table>
| stack| $S$| $d$| $d$| $S$| $d$| $d$| $d$| $S$
| $-$| $S$| $-$| $S$| $-$| $S$| $-$| $S$| $-$|

<table>
<thead>
<tr>
<th>read</th>
<th>$c$</th>
<th>$d$</th>
<th>$d$</th>
<th>$d$</th>
</tr>
</thead>
<tbody>
<tr>
<td>state</td>
<td>$w$</td>
<td>$-$</td>
<td>$w$</td>
<td>$-$</td>
</tr>
<tr>
<td>stack</td>
<td>$S$</td>
<td>$S$</td>
<td>$d$</td>
<td>$d$</td>
</tr>
<tr>
<td>$-$</td>
<td>$d$</td>
<td>$d$</td>
<td>$d$</td>
<td>$d$</td>
</tr>
<tr>
<td>$-$</td>
<td>$S$</td>
<td>$-$</td>
<td>$S$</td>
<td>$-$</td>
</tr>
</tbody>
</table>