

WEEK-3: Design Predictive parser for a given language.

```
#include <stdio.h>
#include <string.h>

char prol[7][10] = { "S", "A", "A", "B", "B", "C", "C" };
char pror[7][10] = { "A", "Bb", "Cd", "aB", "@", "Cc", "@" };
char prod[7][10] = { "S->A", "A->Bb", "A->Cd", "B->aB", "B->@", "C->Cc", "C->@" };
char first[7][10] = { "abcd", "ab", "cd", "a@", "@", "c@", "@" };
char follow[7][10] = { "$", "$", "$", "a$", "b$", "c$", "d$" };
char table[5][6][10];

int numr(char c)
{
    switch (c)
    {
        case 'S':
            return 0;

        case 'A':
            return 1;

        case 'B':
            return 2;

        case 'C':
            return 3;

        case 'a':
            return 0;

        case 'b':
            return 1;

        case 'c':
            return 2;

        case 'd':
            return 3;

        case '$':
            return 4;
    }

    return (2);
}

int main()
{
    int i, j, k;

    for (i = 0; i < 5; i++)
        for (j = 0; j < 6; j++)
            strcpy(table[i][j], " ");

    printf("The following grammar is used for Parsing Table:\n");
```

```

    for (i = 0; i < 7; i++)
        printf("%s\n", prod[i]);

    printf("\nPredictive parsing table:\n");

    fflush(stdin);

    for (i = 0; i < 7; i++)
    {
        k = strlen(first[i]);
        for (j = 0; j < 10; j++)
            if (first[i][j] != '@')
                strcpy(table[numr(prol[i][0]) + 1][numr(first[i][j]) + 1],
prod[i]);
    }

    for (i = 0; i < 7; i++)
    {
        if (strlen(pror[i]) == 1)
        {
            if (pror[i][0] == '@')
            {
                k = strlen(follow[i]);
                for (j = 0; j < k; j++)
                    strcpy(table[numr(prol[i][0]) + 1][numr(follow[i][j]) +
1], prod[i]);
            }
        }
    }

    strcpy(table[0][0], " ");
    strcpy(table[0][1], "a");
    strcpy(table[0][2], "b");
    strcpy(table[0][3], "c");
    strcpy(table[0][4], "d");
    strcpy(table[0][5], "$");
    strcpy(table[1][0], "S");
    strcpy(table[2][0], "A");
    strcpy(table[3][0], "B");
    strcpy(table[4][0], "C");

    printf("\n-----\n");

    for (i = 0; i < 5; i++)
        for (j = 0; j < 6; j++)
        {
            printf("%-10s", table[i][j]);
            if (j == 5)

```

```

printf("\n-----\n");
}
}

```

The screenshot shows a terminal window with the following content:

```

vnrvjlet@vnrvjlet-HP-ProDesk-400-G7-Microtower-PC: ~/Desktop/kk
vnrvjlet@vnrvjlet-HP-ProDesk-400-G7-Microtower-PC:~/Desktop/kk$ cc week3.c
vnrvjlet@vnrvjlet-HP-ProDesk-400-G7-Microtower-PC:~/Desktop/kk$ ./a.out
The following grammar is used for Parsing Table:
S->A
A->Bb
A->Cd
B->aB
B->@
C->Cc
C->@

Predictive parsing table:
-----
      a      b      c      d      $
-----
S      S->A    S->A    S->A    S->A
-----
A      A->Bb    A->Bb    A->Cd    A->Cd
-----
B      B->aB    B->@
-----
C      C->@    C->@    C->@
-----
vnrvjlet@vnrvjlet-HP-ProDesk-400-G7-Microtower-PC:~/Desktop/kk$

```