IT250 – AUTOMATA & COMPILER DESIGN

ASSIGNMENT 9

Name: **Sachin Prasanna**
Roll No.: **211IT058**

Note: Since my roll number is 58, the production rules given to me were:

P → QR;
R→ int
R→ float
S → R,id
S → id
S → (P)

Hence, I have used these production rules in my code to parse the inputted string.

Convention: I have taken the **P** non terminal as the start symbol because by convention, the non-terminal on the left side of the first production rule is taken as the start symbol.

Code Written:

```c
#include <stdio.h>
#include <stdbool.h>
#include <ctype.h>
#include <string.h>
#include <stdlib.h>

typedef struct Production {
    int size;
    char endChar;
    char arr[20];
} Production;

typedef struct Productions {
    int size;
    Production array[20];
    char termStart;
} Productions;

void newProductionSet(Productions *p){
    p->size = 0;
}

void addProduction(Productions *p, char endChar, char *buffer){
    int i;
    p->size++;

    for (i = 0; buffer[i] != '\0' && buffer[i] != '\n'; i++) {
        p->array[p->size - 1].arr[i] = buffer[i];
    }

    p->array[p->size - 1].size = i;
    p->array[p->size - 1].endChar = endChar;
}

void printProductions(Production *p){
    for (int i = 0; i < p->size; i++) {
        printf("%c", p->arr[i]);
    }
}

void shift(int *top, char *st, char *buffer, int *ctr, char ch){
```

```
        printf("shift\n");
        st[*top] = ch;
        st[(*top)++ + 1] = '\0';
        (*ctr)++;

        if (buffer[*ctr] == '\n' || buffer[*ctr] == '\0') ch = '$';
        else ch = buffer[*ctr];
}

void reduce(Productions *set, char *buffer, char *st, int *top, int *ctr) {
    int prod;
    int flag = 1;

    while (flag) {
        for (prod = 0; prod < set->size; prod++) {
            flag = 1;
            int k = set->array[prod].size - 1;

            for (int j = *top - 1; j >= 0 && k >= 0; j--, k--) {
                if (set->array[prod].arr[k] != st[j]) {
                    flag = 0;
                    break;
                }
            }
            if (k != -1) flag = 0;
            if (flag) break;
        }

        if (flag) {
            st[*top] = '\0';
            printf("%s\t\t\t%s\t\t\t", st, buffer + *ctr);
            printf("reduce by ");
            printf(" %c -> ", set->array[prod].endChar);
            printProductions(&set->array[prod]);
            printf("\n");

            for (int i = 0; i < set->array[prod].size; i++) {
                (*top)--;
            }

            st[*top] = set->array[prod].endChar;
            st[*top + 1] = '\0';
            (*top)++;

        } else break;
```

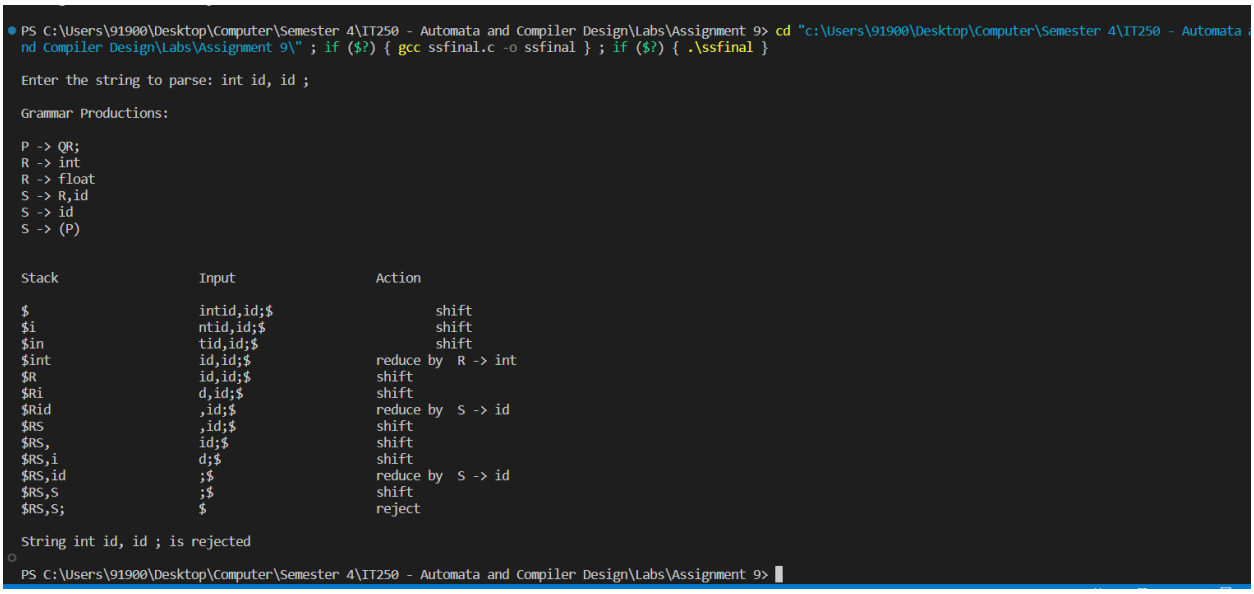
```
    }
}


bool shiftReduceParser(Productions *set, char *buffer){

    char st[1000];
    int top = 0, ctr = 0;
    char ch;

    st[top++] = '$';

    printf("Stack\t\t\tInput\t\t\tAction\n\n");

    while (true) {

        if (buffer[ctr] == '\n' || buffer[ctr] == '\0') ch = '$';
        else ch = buffer[ctr];
        st[top] = '\0';

        printf("%s\t\t\t%s\t\t\t", st, buffer + ctr);

        if (ch == '$'){
            if(top == 2 && st[1] == set->termStart) {
                printf("accepted\n");
                return true;
            }
            else if (top != 1 || st[0] != set->termStart) {
                printf("top = %d\n", top);
                printf("st[0] = %c\n", st[0]);
                printf("reject\n");
                return false;
            }
            else {
                printf("accepted\n");
                return true;
            }
        }

        shift(&top, st, buffer, &ctr, ch);
        //If possible, then reduce
        reduce(set, buffer, st, &top, &ctr);
    }
    return false;
}
```

```
void removeSpaces(char* str){
    int i, j;
    for (i = 0, j = 0; str[i] != '\0'; i++)
    {
        if (!isspace(str[i]))
            str[j++] = str[i];
    }
    str[j] = '\0';
}


int main()
{
    char ch;
    char temp[1000], show[1000];

    // Inserting Productions given in the question
    Productions g;
    newProductionSet(&g);

    char prod1[5] = {'Q','R',';','\0'};
    char prod2[5] = {'i', 'n', 't', '\0'};
    char prod3[8] = {'f', 'l', 'o', 'a', 't', '\0'};
    char prod4[5] = {'R', ',', 'i', 'd', '\0'};
    char prod5[5] = {'i', 'd', '\0'};
    char prod6[5] = {'(', 'P', ')', '\0'};

    addProduction(&g, 'P', prod1);
    addProduction(&g, 'R', prod2);
    addProduction(&g, 'R', prod3);
    addProduction(&g, 'S', prod4);
    addProduction(&g, 'S', prod5);
    addProduction(&g, 'S', prod6);

    g.termStart = 'P';

    printf("\n");

    printf("Enter the string to parse: ");

    scanf(" %[^\n]", temp);

    strcpy(show, temp);
```

```
    removeSpaces(temp);

    int len = strlen(temp);

    printf("len = %d\n", len);

    printf("temp = %s\n", temp);

    if (len > 0 && temp[len - 1] != '$') {
        temp[len] = '$';
        temp[len + 1] = '\0';
    } else {
    strcat(temp, "$");
    }

    printf("temp = %s\n", temp);

    printf("\nGrammar Productions:\n\n");

    for (int i = 0; i < g.size; i++) {
        printf("%c -> ", g.array[i].endChar);
        printProductions(&g.array[i]);
        printf("\n");
    }

    printf("\n\n");

    if (shiftReduceParser(&g, temp)) printf("\nString %s is accepted\n", show);
    else printf("\nString %s is rejected\n\n", show);

    return 0;
}
```

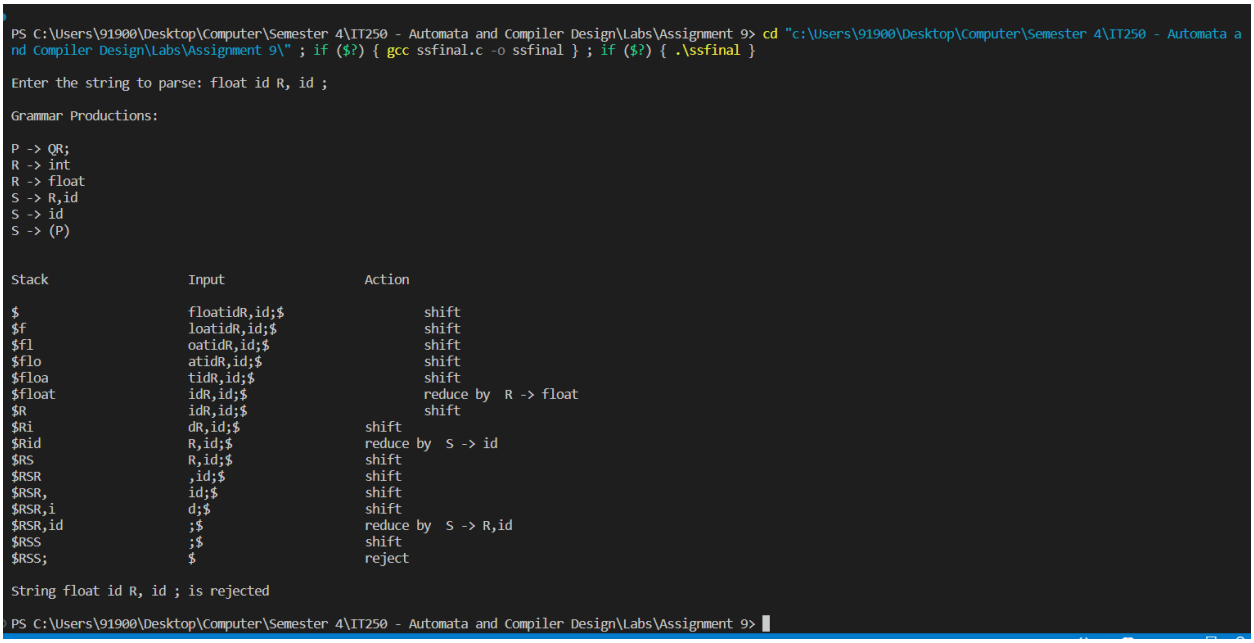
Outputs:

1) Input: int id , id ;



```
PS C:\Users\91900\Desktop\Computer\Semester 4\IT250 - Automata and Compiler Design\Labs\Assignment 9> cd "c:\Users\91900\Desktop\Computer\Semester 4\IT250 - Automata a
nd Compiler Design\Labs\Assignment 9\" ; if ($?) { gcc ssfinal.c -o ssfinal } ; if ($?) { .\ssfinal }

Enter the string to parse: int id, id ;

Grammar Productions:

P -> QR;
R -> int
R -> float
S -> R,id
S -> id
S -> (P)


Stack                   Input                   Action

$                       intid,id;$                      shift
$i                      ntid,id;$                       shift
$in                     tid,id;$                        shift
$int                    id,id;$                 reduce by  R -> int
$R                      id,id;$                 shift
$Ri                     d,id;$                  shift
$Rid                    ,id;$                   reduce by  S -> id
$RS                     ,id;$                   shift
$RS,                    id;$                    shift
$RS,i                   d;$                     shift
$RS,id                  ;$                      reduce by  S -> id
$RS,S                   ;$                      shift
$RS,S;                  $                       reject

String int id, id ; is rejected

PS C:\Users\91900\Desktop\Computer\Semester 4\IT250 - Automata and Compiler Design\Labs\Assignment 9>
```

2) Input: float id R , id ;



```
PS C:\Users\91900\Desktop\Computer\Semester 4\IT250 - Automata and Compiler Design\Labs\Assignment 9> cd "c:\Users\91900\Desktop\Computer\Semester 4\IT250 - Automata a
nd Compiler Design\Labs\Assignment 9\" ; if ($?) { gcc ssfinal.c -o ssfinal } ; if ($?) { .\ssfinal }

Enter the string to parse: float id R, id ;

Grammar Productions:

P -> QR;
R -> int
R -> float
S -> R,id
S -> id
S -> (P)


Stack                   Input                   Action

$                       floatidR,id;$                   shift
$f                      loatidR,id;$                    shift
$fl                     oatidR,id;$                     shift
$flo                    atidR,id;$                      shift
$floa                   tidR,id;$                       shift
$float                  idR,id;$                        reduce by  R -> float
$R                      idR,id;$                        shift
$Ri                     dR,id;$                 shift
$Rid                    R,id;$                  reduce by  S -> id
$RS                     R,id;$                  shift
$RSR                    ,id;$                   shift
$RSR,                   id;$                    shift
$RSR,i                  d;$                     shift
$RSR,id                 ;$                      reduce by  S -> R,id
$RSS                    ;$                      shift
$RSS;                   $                       reject

String float id R, id ; is rejected

PS C:\Users\91900\Desktop\Computer\Semester 4\IT250 - Automata and Compiler Design\Labs\Assignment 9>
```

3) **Input:** int id id int;

```
PS C:\Users\91900\Desktop\Computer\Semester 4\IT250 - Automata and Compiler Design\Labs\Assignment 9> cd "c:\Users\91900\Desktop\Computer\Semester 4\IT250 - Automata a
nd Compiler Design\Labs\Assignment 9\" ; if ($?) { gcc ssfinal.c -o ssfinal } ; if ($?) { .\ssfinal }

Enter the string to parse: int id id int;

Grammar Productions:

P -> QR;
R -> int
R -> float
S -> R,id
S -> id
S -> (P)


Stack                   Input                   Action

$                       intididint;$                    shift
$i                      ntididint;$                     shift
$in                     tididint;$                      shift
$int                    ididint;$                       reduce by  R -> int
$R                      ididint;$                       shift
$Ri                     didint;$                        shift
$Rid                    idint;$                 reduce by  S -> id
$RS                     idint;$                 shift
$RSi                    dint;$                  shift
$RSid                   int;$                   reduce by  S -> id
$RSS                    int;$                   shift
$RSSi                   nt;$                    shift
$RSSin                  t;$                     shift
$RSSint                 ;$                      reduce by  R -> int
$RSSR                   ;$                      shift
$RSSR;                  $                       reject

String int id id int; is rejected

PS C:\Users\91900\Desktop\Computer\Semester 4\IT250 - Automata and Compiler Design\Labs\Assignment 9>
```

4) **Input:** float R S;

```
PS C:\Users\91900\Desktop\Computer\Semester 4\IT250 - Automata and Compiler Design\Labs\Assignment 9> cd "c:\Users\91900\Desktop\Computer\Semester 4\IT250 - Automata a
nd Compiler Design\Labs\Assignment 9\" ; if ($?) { gcc ssfinal.c -o ssfinal } ; if ($?) { .\ssfinal }

Enter the string to parse: float R S;

Grammar Productions:

P -> QR;
R -> int
R -> float
S -> R,id
S -> id
S -> (P)


Stack                   Input                   Action

$                       floatRS;$                       shift
$f                      loatRS;$                        shift
$fl                     oatRS;$                 shift
$flo                    atRS;$                  shift
$floa                   tRS;$                   shift
$float                  RS;$                    reduce by  R -> float
$R                      RS;$                    shift
$RR                     S;$                     shift
$RRS                    ;$                      shift
$RRS;                   $                       reject

String float R S; is rejected

PS C:\Users\91900\Desktop\Computer\Semester 4\IT250 - Automata and Compiler Design\Labs\Assignment 9>
```

****