IT250 – AUTOMATA & COMPILER DESIGN

ASSIGNMENT 10

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

Code Written:

```
class Stack:
    def __init__(self):
        self.__storage = []

    def top(self):
        return self.__storage[len(self.__storage) - 1]

    def push(self,p):
        self.__storage.append(p)

    def length(self):
        return len(self.__storage)

    def pop(self):
        return self.__storage.pop()

    def __str__(self):
        return 'stack ${}]'.format(', '.join([ str(i) for i in
reversed(self.__storage) ]))

def initialise(storer,term):
    temp_array = storer
    for item in temp_array :
```

```
        index = item[1].index('.')
        if(index<(len(item[1])-1) and item[1][index+1] in term):
            for production in term[item[1][index+1]]:
                if( [item[1][index+1],str('.')+str(production)] not in
temp_array):
                    temp_array.append([item[1][index+1],str('.')+str(production)]
)
    return temp_array

status = []
storer = []
array_r = []
accept = -1
mpp = dict()
tbb = []
T = []
NT = dict()

def cannonical(start, nonTer, ter):
    storer.append(initialise([['start', '.' + start + '$']], nonTer))
    ter += list(nonTer.keys())
    for conroller in storer:
        for grammar in ter:
            if grammar == '$':
                continue
            goto_flag, goto1_flag, shift_flag, shift1_flag, reduce_flag = False,
False, False, False, False
            close = []
            for item in conroller:
                if item[1].index('.') < (len(item[1]) - 1) and
item[1][item[1].index('.') + 1] == grammar:
                    close.append([item[0], item[1][:item[1].index('.')] + grammar
+ '.' + item[1][item[1].index('.') + 2:]])
            l = initialise(close, nonTer)
            if len(l) == 0:
                continue
            if grammar in nonTer.keys():
                goto1_flag = True
            else:
                shift1_flag = True
            if l not in storer:
                if goto1_flag:
                    status.append(['g', storer.index(conroller) + 1, len(storer)
+ 1, grammar])
                    goto_flag = True
```

```
                elif shift1_flag:
                    shift_flag = True
                    status.append(['s', storer.index(conroller) + 1, len(storer)
+ 1, grammar])
                storer.append(l)
            else:
                if goto1_flag:
                    goto_flag = True
                    status.append(['g', storer.index(conroller) + 1,
storer.index(l) + 1, grammar])
                elif shift1_flag:
                    shift_flag = True
                    status.append(['s', storer.index(conroller) + 1,
storer.index(l) + 1, grammar])


def reduction(rule,accept,start):
    s = ['start',start+'.$']
    for pp in storer:
        if(s in pp):
            accept = storer.index(pp)
        for item in pp:
            if( item in rule):
                array_r[storer.index(pp)].append(rule.index(item))

    return accept

def parsed(ter):
    for i in status:
        tbb[i[1]-1][mpp[i[3]]] = i[0]+str(i[2]-1)

    tbb[accept][mpp['$']] = 'a'

    for i in array_r:
        if(len(i)>0):
            for j in ter:
                tbb[array_r.index(i)][mpp[j]] = 'r'+str(i[0])

def LRParser(rule, string):
    is_accepted = False
    stack = Stack()
    stack.push('0')
    index = 0
    print("Stack\t\t\tInput\t\t\tAction\n")
    while index < len(string):
```

```
        stack_content = ''.join(str(stack).split()[1:-1]).replace(",", "")
        input_remaining = string[index:]
        print(f"{stack_content:<15}", end='\t\t')
        print(f"{input_remaining:<15}", end='\t\t')
        c = tbb[int(stack.top())][mpp[string[index]]][0]
        if c == 'a':
            is_accepted = True
            break
        pt = tbb[int(stack.top())][mpp[string[index]]][1:]
        if pt:
            pt = int(pt)
            if c == 'r':
                reduction_length = len(rule[pt][1])
                reduction_length *= 2
                reduction_length -= 2
                if reduction_length >= stack.length():
                    break
                else:
                    for i in range(reduction_length):
                        stack.pop()
                    top = int(stack.top())
                    stack.push(rule[pt][0])
                    stack.push(tbb[top][mpp[stack.top()]][1:])
                    print(f"Reduce: r{pt}")
            else:
                stack.push(string[index])
                stack.push(str(pt))
                index += 1
                print(f"Shift: s{pt}")
        else:
            break
    if is_accepted:
        print("Accepted")
    else:
        print("Not Accepted")
    return is_accepted


T = ['b','c', '$']
n = 2
NT["S'"] = ['S']
NT['S'] = ['AS','b']
NT['A'] = ['SA', 'c']

S  = 'S'
```

```
cannonical(S,NT,T)

rule = []
accept = -1

for i in NT.keys():
    for j in NT[i]:
        rule.append([i,j+str('.')])

array_r = [ [] for i in range(len(storer)) ]
accept = reduction(rule,accept,S)

sym = []
sym += T

for count , i in enumerate(sym):
    mpp[i] = count

tbb = [ ['-' for i in range(len(sym))] for j in range(len(storer)) ]

for i in NT.keys():
    T.remove(i)

parsed(T)

file_path = "file1.txt"

with open(file_path, 'r') as file:
    string = file.read().strip()

string+='$'

print("Input 1 output:\n")

if(LRParser(rule,string)):
    print("\n" + string[:-1] + " is accepted\n")
else:
    print("\n" + string[:-1] + " is not accepted\n")

file_path_2 = "file2.txt"

with open(file_path_2, 'r') as file:
    string_1 = file.read().strip()
```

```
string_1 +='$'

print("\nInput 2 output:\n")

if(LRParser(rule,string_1)):
    print("\n" + string_1[:-1] + " is accepted\n")
else:
    print("\n" + string_1[:-1] + " is not accepted\n")
```

Input files:

I made two different input files for 2 inputs and read them from the files and sent them for parsing.

The outputs obtained is thus shown below.

String 1 (in file1.txt): cccbbcbcbcb
String 2 (in file2.txt): cbbbbcbcbbbbc

Outputs:

```
PS C:\Users\91900\Desktop\Computer\Semester 4\IT250 - Automata and Compiler Design\Labs\Assignment 10> python -u "c:\Users\91900\Desktop\Computer\Semester 4\IT250 - Automata and Compiler Design\Labs\Assignment 10\a2.py"
Input 1 output:

Stack                   Input                   Action

                        cccbbcbcbcb$            Shift: s2
$2c                     ccbbcbcbcb$             Reduce: r3
$4A                     ccbbcbcbcb$             Shift: s2
$2c4A                   cbbcbcbcb$              Reduce: r3
$4A4A                   cbbcbcbcb$              Shift: s2
$2c4A4A                 bbcbcbcb$               Reduce: r3
$4A4A4A                 bbcbcbcb$               Shift: s1
$1b4A4A4A               bcbcbcb$                Reduce: r1
$7S4A4A4A               bcbcbcb$                Reduce: r0
$7S4A4A                 bcbcbcb$                Reduce: r0
$7S4A                   bcbcbcb$                Reduce: r0
$3S                     bcbcbcb$                Shift: s1
$1b3S                   cbcbcb$                 Reduce: r1
$5S3S                   cbcbcb$                 Shift: s2
$2c5S3S                 bcbcb$                  Reduce: r3
$6A5S3S                 bcbcb$                  Reduce: r2
$6A3S                   bcbcb$                  Reduce: r2
$4A                     bcbcb$                  Shift: s1
$1b4A                   cbcb$                   Reduce: r1
$7S4A                   cbcb$                   Reduce: r0
$3S                     cbcb$                   Shift: s2
$2c3S                   bcb$                    Reduce: r3
$6A3S                   bcb$                    Reduce: r2
$4A                     bcb$                    Shift: s1
$1b4A                   cb$                     Reduce: r1
$7S4A                   cb$                     Reduce: r0
$3S                     cb$                     Shift: s2
$2c3S                   b$                      Reduce: r3
$6A3S                   b$                      Reduce: r2
$4A                     b$                      Shift: s1
$1b4A                   $                       Reduce: r1
$7S4A                   $                       Reduce: r0
$3S                     $                       Accepted

cccbbcbcbcb is accepted
```

```
Input 2 output:

Stack                   Input                   Action

                        cbbbbcbcbbbbc$          Shift: s2
$2c                     bbbbcbcbbbbc$           Reduce: r3
$4A                     bbbbcbcbbbbc$           Shift: s1
$1b4A                   bbbcbcbbbbc$            Reduce: r1
$7S4A                   bbbcbcbbbbc$            Reduce: r0
$3S                     bbbcbcbbbbc$            Shift: s1
$1b3S                   bbcbcbbbbc$             Reduce: r1
$5S3S                   bbcbcbbbbc$             Shift: s1
$1b5S3S                 bcbcbbbbc$              Reduce: r1
$5S5S3S                 bcbcbbbbc$              Shift: s1
$1b5S5S3S               cbcbbbbc$               Reduce: r1
$5S5S5S3S               cbcbbbbc$               Shift: s2
$2c5S5S5S3S             bcbbbbc$                Reduce: r3
$6A5S5S5S3S             bcbbbbc$                Reduce: r2
$6A5S5S3S               bcbbbbc$                Reduce: r2
$6A5S3S                 bcbbbbc$                Reduce: r2
$6A3S                   bcbbbbc$                Reduce: r2
$4A                     bcbbbbc$                Shift: s1
$1b4A                   cbbbbc$                 Reduce: r1
$7S4A                   cbbbbc$                 Reduce: r0
$3S                     cbbbbc$                 Shift: s2
$2c3S                   bbbbc$                  Reduce: r3
$6A3S                   bbbbc$                  Reduce: r2
$4A                     bbbbc$                  Shift: s1
$1b4A                   bbbc$                   Reduce: r1
$7S4A                   bbbc$                   Reduce: r0
$3S                     bbbc$                   Shift: s1
$1b3S                   bbc$                    Reduce: r1
$5S3S                   bbc$                    Shift: s1
$1b5S3S                 bc$                     Reduce: r1
$5S5S3S                 bc$                     Shift: s1
$1b5S5S3S               c$                      Reduce: r1
$5S5S5S3S               c$                      Shift: s2
$2c5S5S5S3S             $                       Reduce: r3
$6A5S5S5S3S             $                       Reduce: r2
$6A5S5S3S               $                       Reduce: r2
$6A5S3S                 $                       Reduce: r2
$6A3S                   $                       Reduce: r2
$4A                     $                       Not Accepted

cbbbbcbcbbbbc is not accepted
```

****