ASCII Dancer…

Project Description:

In this project, we created a dancing ASCII man. A series of commands is given which will indicate either the moves the man makes or the things the man says. If an input line starts with the command "say ", program will output the rest of the line. The program would output the mirror image if the man was facing backwards and given the same commands, or if the man is facing forwards.

The command "turn" causes the ASCII man to change the way he is facing, for example, facing backwards if he was facing forwards, or facing forwards if he was facing backwards. The input begins with an integer T giving the number of test cases. Each test case begins with an integer D that gives the number of commands in the dance.

Introduction:

ASCII Dancer is developed using Assembly Language on Visual Basic 2019. ASCII man is a character. ASCII man dances on the commands given by the user, each set of commands represent a test case. There is a limitation for dance commands, in this project there are total 12 commands from which the user can make the ASCII character dance. One of the 12 commands determines what the ASCII man is going to do next or what the man is saying and this command start with “say ” and then user can write rest of the line.

Development of Game:

To develop this project, the entire operation has been divided into the following step:

1. Creating Array which store series of commands.
2. Dance Moves
3. Validation and Verification
4. Building Project

Creating Array:

The commands that represents the dance moves or what the ASCII man is saying both are stored in the same variable of size byte, in this case the variable is “Say” and it is initialized like this: Say BYTE MAXLEN * 30 DUP(?)

MAXLEN is an immediate value and this is initialized with 60 which means that total command can be stored and each command can contain 30 characters, not more than that.

```
MOV Command, EAX
MOV J, 0
FirstForLoop:
MOV EAX, Command
CMP J, EAX
JE OutOfFirstForLoop
MOV EAX, MAXLEN
MUL J
LEA EDX, Say[EAX]
MOV ECX, MAXLEN
Call ReadString
NC CountLine
INC J
JMP FirstForLoop
OutOfFirstForLoop:
```

The code in the screenshot represents that how the command are read from the user.

Dance Moves:

There are total 11 commands from which the user can make the ASCII man dance and there is one command from which the user can define the upcoming moves of the ASCII man or what the ASCII man is about to say.

- Following are the Dance Moves:
    1. "left hand to head"
    2. "left hand to hip"
    3. "left hand to start"
    4. "left leg in"
    5. "left leg out"
    6. "right hand to head"
    7. "right hand to hip"
    8. "right hand to start"
    9. "right leg in"
    10. "right leg out"
    11. "turn"
- Other command is “say [words]”

```
CMP Say[EAX + 0], 's'
JNE DANCEMOVES
CMP Say[EAX + 1], 'a'
JNE DANCEMOVES
CMP Say[EAX + 2], 'y'
JNE DANCEMOVES
CMP Say[EAX + 3], ' '
JNE DANCEMOVES
ADD EAX, 4
LEA EDX, Say[EAX]
Call WriteString
Call Crlf
```

This screenshot represents that if the program finds that the first four characters of the command is “say “ then the “say ” is eliminated from the command and the rest of the command is displayed and then it moves to next command.

```
PUSH OFFSET Say
PUSH EAX
Call CheckRightLegIn
Call CheckRightLegOut
Call CheckLeftHandToHead
Call CheckRightHandToHip
Call CheckLeftHandToHip
Call CheckRightHandToHead
Call CheckTurn
Call CheckRightHandToStart
Call CheckLeftHandToStart
Call CheckLeftLegIn
Call CheckLeftLegOut
```

This screenshot represents that the command did not start with “say ” and now it is checking for the corresponding dance move. First of all the command is loaded in stack and each procedure is called one by one to check for the dance move.

Validation and Verification:

```
PrintInvalid:
        MOV EDX, OFFSET InvalidPrompt      ;;;;Display Invalid Prompt
        Call WriteString
        Call Crlf

        ADD CountLine, 2
    JMP GoBack
```

If the user enters invalid command then the program runs this segment of code given in the above screenshot and the program displays “Invalid Command!”

```
BothLegIn:
    ADD CountLine, 2
            MOV LegChecker, 0
            MOV EDX, OFFSET BothLegException
            Call WriteString
            Call Crlf
        JMP GoBack
```

The program is also checking every time whether user has entered right leg in then left leg in or vice versa then according to our project this condition is also invalid and the program displays an exception on the console, “Both Legs Cannot Be In!” and does not display the ASCII man on that second command but then rest of the code is correctly executed.

Building Project:

The instructions of this program are written in main while loop and after each test case the variables and registers hold the values of the previous test case, so to resolve this, after each test case we have given the default values in the variables and registers again so each test case if executed properly.

```
MOV TestCase, EAX
        _While:
                        MOV RH, '/'
                        MOV LH, '\'
                        MOV RF, '/'
                        MOV LF, '\'
                        CMP TestCase, 0
                        JLE _EndWhile
                        Call ReadInt
```

The screenshot above represents how the variables are given default values after each test case.

The ASCII man can dance in two directions backward and forward, so in this project we made two procedure, one is ForwardDisplayDancer and the second is BackwardDisplayDancer. The first procedure is executed when the ASCII man is facing forward and that procedure is called to display the ASCII character. The second procedure is executed when the ASCII man is facing backward so this procedure is called to display the ASCII character. The ASCII man changes direction when the user uses the command “turn”.

This project also uses some animation for the ASCII dancer to process the dance commands, for this case a procedure is called “Text_Animation” so the ASCII man process the dance commands with some delay and some animation.