Program 3:

WAP to implement a MOORE MACHINE, where the program generates an output corresponding to an input string given through the console.

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

#define MAX_TRANSISTION 100
#define MAX_VARIABLES 100

int Read(char *str)
{
 int i = 0;
 while (1)
 {
  str[i] = getchar();
  if (str[i] == '\n' || str[i] == '\r')
  {
   str[i] = '\0';
   return i;
  }
  i++;
 }
}

int main()
{

 int tempInt1, tempInt2;
 char ch1, ch2, ch;
 int i = 0, j = 0, k = 0;
 char str1[10] = {0};
 char str2[10] = {0};

 FILE *filePointer = NULL;
 filePointer = fopen("MOORE2.txt", "r");

 if (filePointer == NULL)
 {
  printf("Unable to open MOORE.txt\n");
  return 1;
 }

 int Transistion_Table[MAX_TRANSISTION][MAX_VARIABLES];
 char OutputTable[MAX_TRANSISTION][10];

 for (i = 0; i < MAX_TRANSISTION; ++i)
```

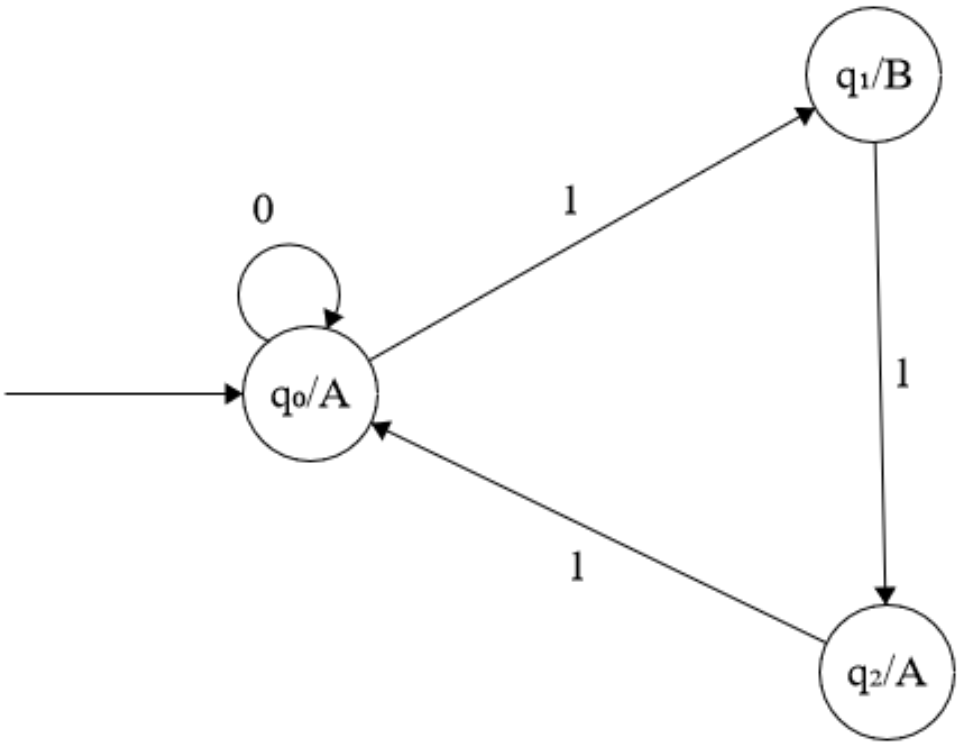
```
{
 for (j = 0; j < MAX_VARIABLES; ++j)
  Transistion_Table[i][j] = 0;
 OutputTable[i][0] = '\0';
}

int initialState = -1;
int currentState = -1;
int numberOfStates = 0;
int numberOfInputs = 0;

fscanf(filePointer, "%d%c", &initialState, &ch1);
printf("Initial state : %d\n", initialState);
fscanf(filePointer, "%d%c", &numberOfInputs, &ch1);
printf("Number of inputs : %d\n", numberOfInputs);

i = j = 0;

while (1)
{
 if (fscanf(filePointer, "%s", str1) == EOF)
  break;
 tempInt1 = atoi(str1);
 Transistion_Table[i][j] = tempInt1;
 j++;
 if (j == numberOfInputs)
 {
  fscanf(filePointer, "%s", str1);
  strcpy(OutputTable[i], str1);
  j = 0, i++;
 }
}
numberOfStates = i;
fclose(filePointer);

printf("Number of states : %d\n", numberOfStates);

printf("Transistion Table\n");
printf("| State | Input(0) | Input(1) |  Output |\n");
for (i = 0; i < numberOfStates; ++i)
{
 printf("| %3d   | ", i);
 for (j = 0; j < numberOfInputs; j++)
 {
```

```
   printf("    %3d    | ", Transistion_Table[i][j]);
  }
  printf("%6s   | \n", OutputTable[i]);
 }

 int flag = 0;
 char inputString[100] = {0};
 char outputString[100] = {0};
 int inputStringLen = 0;
 int outputStringLen = 0;

 while (1)
 {
  outputString[0] = '\0';

  printf("\n-----------------------\nEnter input string ('#' to
exit) : ");
  inputStringLen = Read(inputString);

  if (inputString[0] == '#')
   break;

  printf("Input string : %s, Input string len : %d\n\n",
inputString, inputStringLen);

  currentState = initialState;
  flag = 0;
  for (i = 0; i < inputStringLen; ++i)
  {
   tempInt1 = inputString[i] - '0';

   printf("input string and state transition : %d , q%d / %s -> ",
tempInt1, currentState, OutputTable[currentState]);
   int oldState = currentState;
   currentState = Transistion_Table[currentState][tempInt1];
   strcat(outputString, OutputTable[currentState]);

   if (currentState == -1)
   {
    printf("NO TRANSISTION\n");
    flag = 1;
    break;
   }
   printf("q%d / %c\n", currentState, outputString[i]);
```

```
  }
  outputString[i] = '\0';

  printf("\nOutput of Moore Machine : %s\n", outputString);
 }

 printf("\nExiting ... \n");

 return 0;
}
```

MOORE.txt

```
1  0 2
2  0 1 A
3  -1 2 B
4  -1 0 A
```



Output :

```
Number of inputs : 2
Number of states : 3
Transistion Table
| State | Input(0) | Input(1) |  Output |
|   0   |      0   |      1   |      A  |
|   1   |     -1   |      2   |      B  |
|   2   |     -1   |      0   |      A  |

-----------------------
Enter input string ('#' to exit) : 00100
Input string : 00100, Input string len : 5

input string and state transition : 0 , q0 / A -> q0 / A
input string and state transition : 0 , q0 / A -> q0 / A
input string and state transition : 1 , q0 / A -> q1 / B
input string and state transition : 0 , q1 / B -> NO TRANSISTION

Output of Moore Machine : AAB

-----------------------
Enter input string ('#' to exit) : #

Exiting...
```