NAME:- Sourav Paul
GROUP:- D2
REG. NO:- 20214056
BRANCH:- CSE DEPT.

ASSIGNMENT - 10

Q1 Implement turing machine in C.
Ans:-

```
#include <stdio.h>
#include <string.h>
#define MAX_TAPE_SIZE 1000 // Maximum size of the tape

typedef struct {
char state; // Current state char symbol; // Current symbol
```

```
char newState;
    char newSymbol;
    char direction;
} Transition;
typedef struct {
char tape[MAX_TAPE_SIZE]; int head;

} TuringMachine;
// New state
// New symbol
// Direction to move
// Tape for input
// Head position on the tape

Transition transitions[] = {
// Define your transitions here
// Example: {'A', '0', 'B', '1', 'R'}

};

int numTransitions = sizeof(transitions) / sizeof(Transition);

void executeTransition(TuringMachine *tm, char state, char symbol) { for (int i
= 0; i < numTransitions; i++) {

{

if (transitions[i].state == state && transitions[i].symbol == symbol)

tm->state = transitions[i].newState; tm->tape[tm->head] =
transitions[i].newSymbol; if (transitions[i].direction == 'R') {
} else if (transitions[i].direction == 'L') { tm->head--;

}

break; }

} }

void runTuringMachine(TuringMachine *tm) { tm->state = 'A'; // Initial state
```

```
while (1) {
char state = tm->state;
char symbol = tm->tape[tm->head];
```

```
int main() {
TuringMachine tm;
memset(tm.tape, ' ', sizeof(tm.tape)); // Initialize tape with blank

spaces
tm.head = MAX_TAPE_SIZE / 2; // Initialize head at the middle of

the tape

// Input your initial tape contents here // Example: strcpy(tm.tape, "001001");

runTuringMachine(&tm); return 0;

}
```

Q2. L={WWR | W∈(0+1) * }. Construct a PDA for the language L and write the program in C.

```
#include <stdio.h> #include <string.h> #include <stdbool.h>
// Define constants for PDA transitions

#define TRANSITION_0 '0' #define TRANSITION_1 '1' #define
TRANSITION_X 'X' #define TRANSITION_Y 'Y' #define TRANSITION_Z 'Z'
#define TRANSITION_EMPTY '\0'

// PDA transition function
```

```
*pop = 'Z';
    *push = 'X';
    return true;
} else if (stack_top == 'Z' && input_char == '1') { *pop = 'Z';

*push = 'Y';

return true;
} else if (stack_top == 'X' && input_char == '0') {

    *pop = 'X';
    *push = 'X';
    return true;
} else if (stack_top == 'X' && input_char == '1') { *pop = 'X';
```

```
*push = 'Y';

return true;

    } else if (stack_top == 'Y' && input_char == '0') {
            *pop = 'Y';

            *push = 'X';
            return true;


    } else if (stack_top == 'Y' && input_char == '1') { *pop = 'Y';
    *push = 'Y';
    return true;


    } else if (stack_top == 'X' && input_char == '\0') {
            *pop = 'X';

            *push = 'Z';
            return true;


    } else if (stack_top == 'Y' && input_char == '\0') { *pop = 'Y';
    *push = 'Z';
            return true;

       } else {

            return false;
       }

    }
    // PDA simulation function

    bool isWWRLanguage(char *input) {
```

```
char stack[1000]; // Stack to simulate PDA int top = -1; // Top of the stack

// Initial push to stack
stack[++top] = 'Z';
// Loop through the input string

for (int i = 0; i < strlen(input); i++) { char pop, push;

if (transition(stack[top], input[i], &pop, &push)) {
```

```
// PDA transition is valid, update the stack

top--;
if (push != TRANSITION_EMPTY) {

                stack[++top] = push;
            }
} else {
return false; // PDA transition is invalid

} }

// Check if the stack is empty and input is fully consumed

return top == -1 && input[strlen(input)] == '\0'; }
```

```
int main() {
char input[100];
printf("Enter a string: ");
fgets(input, sizeof(input), stdin);
input[strcspn(input, "\n")] = '\0'; // Remove newline character from input

if (isWWRLanguage(input)) {
printf("'%s' belongs to the language L={WWR | W∈(0+1)*}.\n", input);

} else {
printf("'%s' does not belong to the language L={WWR | W∈(0+1)*}.\n",

input); }

return 0; }
```

Q3). L={an b 3n| n>=1}. Construct a PDA for the language L and write the program in C

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

#define MAX_SIZE 100
// Stack implementation
char stack[MAX_SIZE]; int top = -1;

void push(char ch) {
```

```
stack[++top] = ch;
}
char pop() {
    if (top == -1) {
        return '\0';
    }
    return stack[top--];
}
// PDA function to check if input string is in language L

bool isLanguageL(char* str) { int len = strlen(str);
```

```
{

{

int i = 0;

// Transition function
char currentState = 'q0'; char inputSymbol;
char stackSymbol;

while (i < len) { inputSymbol = str[i++];

if (currentState == 'q0' && inputSymbol == 'a') { push('a');

      currentState = 'q1';
  •    } else if (currentState == 'q1' && inputSymbol == 'a') {
       push('a');


  •    } else if (currentState == 'q1' && inputSymbol == 'b' && pop() == 'a')
            currentState = 'q2';

  •

  •    } else if (currentState == 'q2' && inputSymbol == 'b' && pop() == 'a')
       // Stay at q2
          } else {

  •          return false; // Reject
  •

} }
```

```
if (currentState == 'q2' && pop() == '\0') { return true; // Accept

}

   return false; // Reject
}
int main() {
   char input[MAX_SIZE];
```

```
printf("Enter input string: "); scanf("%s", input);

if (isLanguageL(input)) {
printf("Input string is in language L.\n");
```

```
} else {
printf("Input string is not in language L.\n");
}
return 0; }
```