**Practical No. - 1**

**Aim: - Implement “mkdir” command of DOS in C language.**

**Program:-**

#include <stdio.h>

#include <stdlib.h>

#include <conio.h>

#include <dir.h>

int main()

{

int flag = 1;

char dirName[10];

while(flag!=0)

{

printf("Enter a directory name:\n");

printf("E.g.: D:\\Movies\\\n");

scanf("%s",&dirName);

flag = mkdir(dirName);

if(flag!=0)

printf("Couldn't create the directory. Try another name.\n");

else

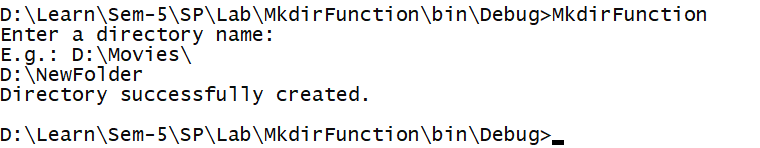
printf("Directory successfully created.\n");

}

return 0;

}

**Output:-**

****

**Practical No. - 2**

**Aim: - Implement “copy” command of DOS in C language.**

**Program:-**

#include <stdio.h>

#include <stdlib.h>

int main()

{

char path[100];

printf("Enter file path to be opened.\n");

scanf("%s",&path);

FILE \*read, \*write;

read = fopen(path,"r");

write = fopen("writeFile.txt","w");

while(!feof(read))

{

char c = fgetc(read);

}

int curpos = ftell(read);

fseek(read,0,0);

while(curpos>0)

{

char c = fgetc(read);

fputc(c,write);

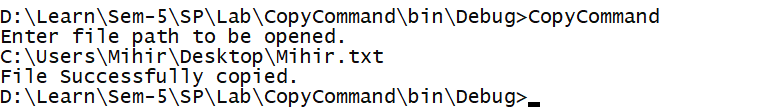
curpos--;

}

printf(“File successfully copied.”);

}

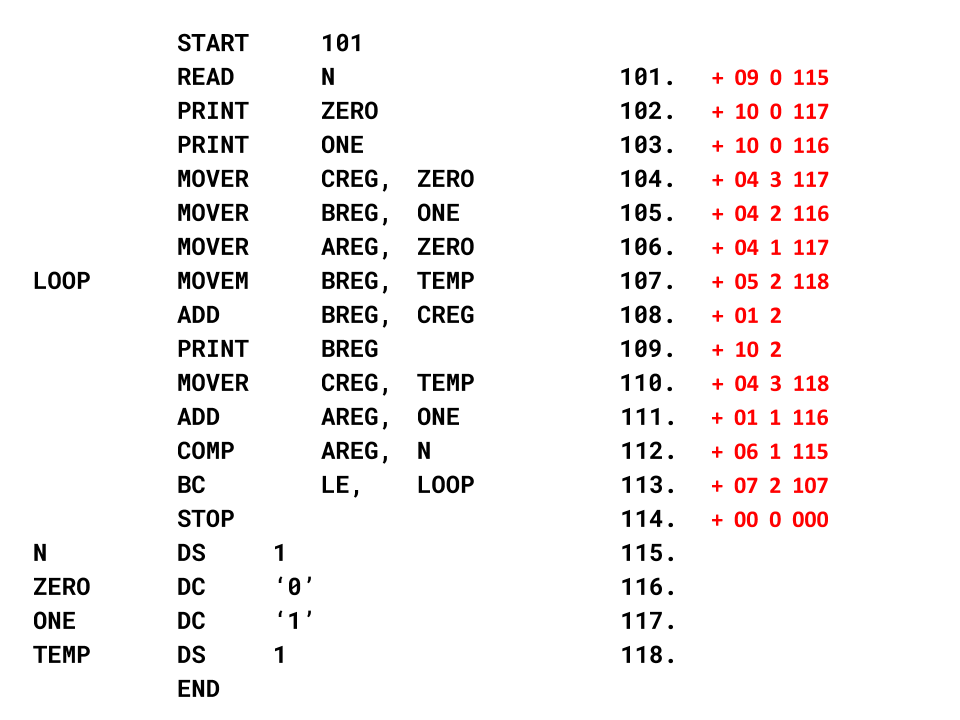
**Output:-**

****

**Practical No. - 3**

**Aim: - Write an assembly language program to print Fibonacci Series.**

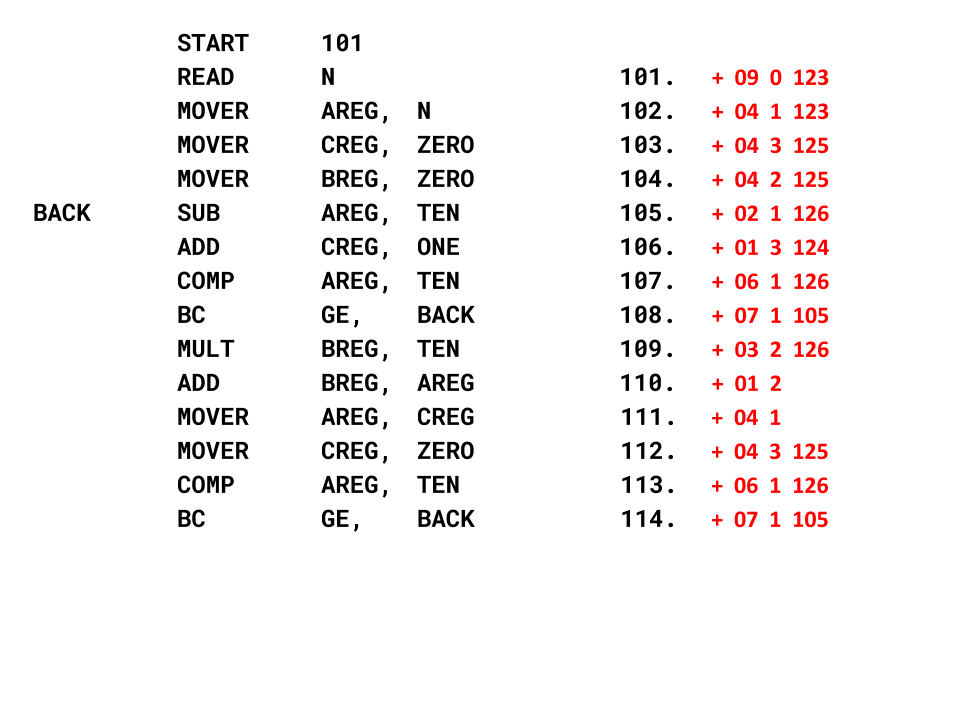
**Program:-**

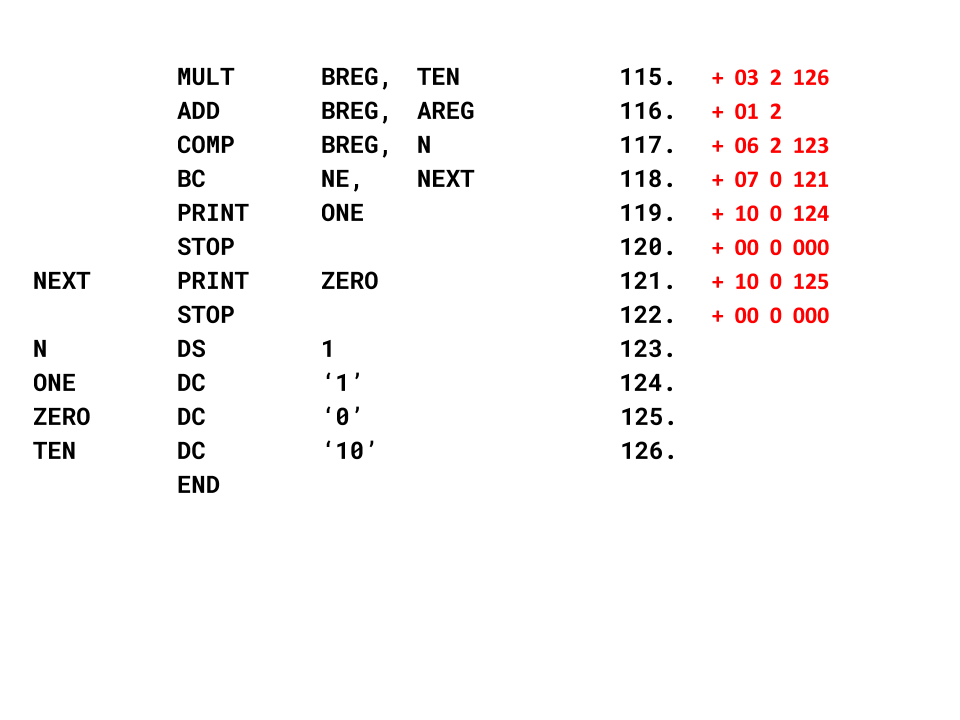
****

**Practical No. - 4**

**Aim: - Write an assembly language program to check if the string is palindrome.**

**Program:-**

****

****

**Practical No. - 5**

**Aim: - Write a program that recognizes the entered string of characters.**

**Program:-**

#include<stdio.h>

#include<conio.h>

int main()

{

char ch,string[100];

int stringIndex = 0;

printf("Enter the text: ");

while(1)

{

ch=fgetc(stdin);

if(ch=='\n')

{

string[stringIndex++] = ch;

ch = fgetc(stdin);

if(ch == '\n')

break;

else

string[stringIndex++] = ch;

}

else

string[stringIndex++] = ch;

}

for(int s = 0; s < stringIndex ;s++)

printf("%c",string[s]);

for(int s = 0; s < stringIndex ;s++)

{

if(string[s] >= 'A' && string[s] <= 'Z')

printf("%c : uppercase.\n",string[s]);

if(string[s] >= 'a' && string[s] <= 'z')

printf("%c : lowercase.\n",string[s]);

switch(string[s])

{

case '+': case '-': case '\*': case '/': case '%': case '=':

printf("%c : operator.\n",string[s]); break;

case '!': case '@': case '#': case '$': case '^':

case '&': case '(': case ')': case '{': case '}':

case '[': case ']': case ',': case '.': case '\\':

case '<': case '>': case '?': case ';': case ':':

case '\'': case '"': case '|': case '`': case '~':

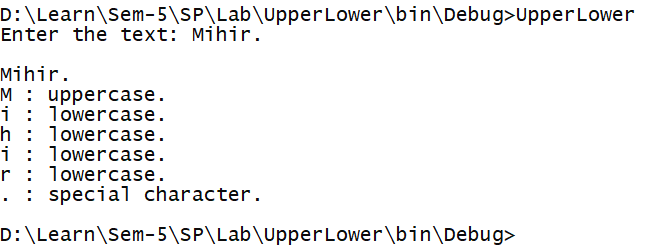
printf("%c : special character.\n",string[s]);

}

}

getch();

}

**Output:-**

**Practical No. - 6**

**Aim: - Write a program that converts infix expression to postfix.**

**Program:-**

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

char stack[100];

int top = -1;

void push(char temp)

{

top++;

stack[top] = temp;

}

char pop()

{

return stack[top--];

}

int getPriority(char operator)

{

switch(operator){

case '+': case '-': return 1; break;

case '\*': case '/': return 2; break;

case '^': return 3; break;

} }

int main()

{

int i;

char infix[100];

printf("Enter infix: ");

scanf("%s",infix);

int length = strlen(infix);

for(i = 0; infix[i] != '\0'; i++ )

{

switch(infix[i])

{

case '(':

push(infix[i]);

break;

case ')':

while( stack[top] != '(')

{

char temp = pop();

printf("%c",temp);

}

pop();

break;

case '+': case '-': case '\*': case '/': case '^':

//check priority

//push if stackPriority is > infix priority

//printf("P1 : %d ",getPriority(stack[top]));

//printf("P2 : %d\n",getPriority(infix[i]));

if(top == -1 || stack[top] == '(')

push(infix[i]);

else if( getPriority(stack[top]) < getPriority(infix[i]) )

push(infix[i]);

else

{

//pop the whole stack until lower priority

while( getPriority(stack[top]) >= getPriority(infix[i]) && stack[top] != '(')

{

char temp = pop();

printf("%c",temp);

}

push(infix[i]);

}

break;

default: printf("%c",infix[i]); break;

}

}

//pop the remaining stack

while(top >= 0)

{

char temp = pop();

printf("%c",temp);

}

}

**Output:-**