Practical No. – 1

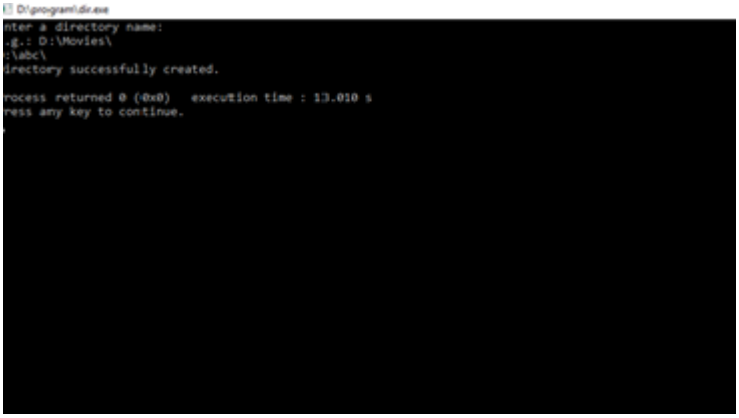
Aim: - Implement “mkdir” command of DOS in C language.

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:


D:\program\dir.exe
nter a directory name:
.g.: D:\Movies\
:\abc\
irectory successfully created.
rocess returned 0 (0x0) execution time : 13.010 s
ress any key to continue.

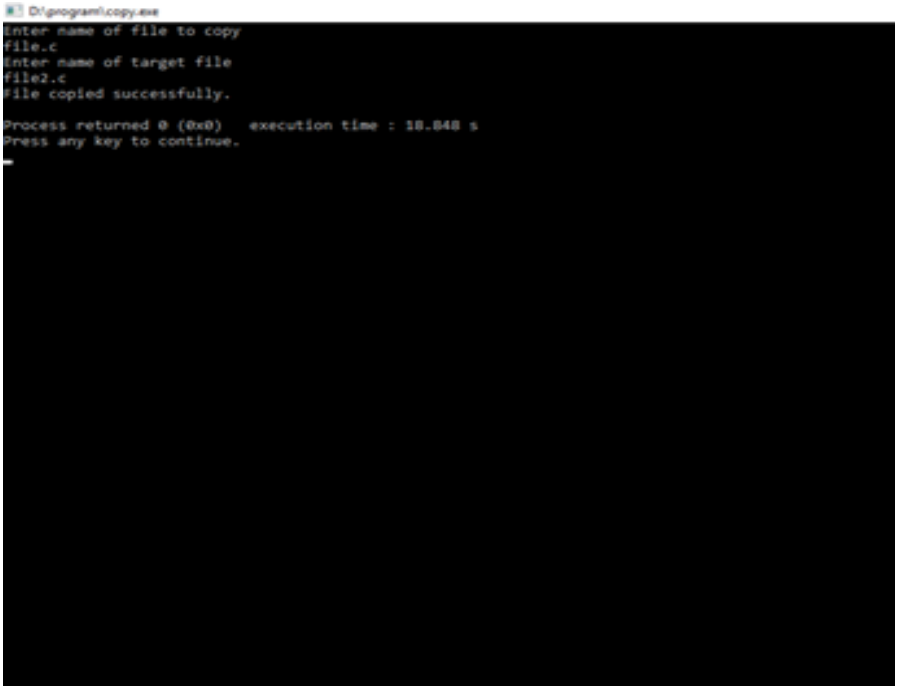
Practical No. – 2

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

```
#include <stdio.h>
#include <stdlib.h>
int main()
{
  char ch, source_file[20], target_file[20];
  FILE *source, *target;
  printf("Enter name of file to copy\n");
  gets(source_file);
  source = fopen(source_file, "r");
  if (source == NULL)
  {
     printf("Press any key to exit...\n");
     exit(EXIT_FAILURE);
   }
     printf("Enter name of target file\n");
     gets(target_file);
     target = fopen(target_file, "w");
   if (target == NULL)
   {
       fclose(source);
       printf("Press any key to exit...\n");
       exit(EXIT_FAILURE);
    }
   while ((ch = fgetc(source)) != EOF)
  fputc(ch, target);
  printf("File copied successfully.\n");
  fclose(source);
  fclose(target);
```

```
  return 0;
}
```

Output:



Practical No. - 3

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

Program:-

```
        START   101
        READ    N                       101.  + 09 0 115
        PRINT   ZERO                    102.  + 10 0 117
        PRINT   ONE                     103.  + 10 0 116
        MOVER   CREG,  ZERO             104.  + 04 3 117
        MOVER   BREG,  ONE              105.  + 04 2 116
        MOVER   AREG,  ZERO             106.  + 04 1 117
LOOP    MOVEM   BREG,  TEMP             107.  + 05 2 118
        ADD     BREG,  CREG             108.  + 01 2
        PRINT   BREG                    109.  + 10 2
        MOVER   CREG,  TEMP             110.  + 04 3 118
        ADD     AREG,  ONE              111.  + 01 1 116
        COMP    AREG,  N                112.  + 06 1 115
        BC      LE,    LOOP             113.  + 07 2 107
        STOP                            114.  + 00 0 000
N       DS    1                         115.
ZERO    DC    '0'                       116.
ONE     DC    '1'                       117.
TEMP    DS    1                         118.
        END
```

Practical No. – 4

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

Program:-

```
        START   101
        READ    N                   101.  + 09 0 123
        MOVER   AREG,  N            102.  + 04 1 123
        MOVER   CREG,  ZERO         103.  + 04 3 125
        MOVER   BREG,  ZERO         104.  + 04 2 125
BACK    SUB     AREG,  TEN          105.  + 02 1 126
        ADD     CREG,  ONE          106.  + 01 3 124
        COMP    AREG,  TEN          107.  + 06 1 126
        BC      GE,    BACK         108.  + 07 1 105
        MULT    BREG,  TEN          109.  + 03 2 126
        ADD     BREG,  AREG         110.  + 01 2
        MOVER   AREG,  CREG         111.  + 04 1
        MOVER   CREG,  ZERO         112.  + 04 3 125
        COMP    AREG,  TEN          113.  + 06 1 126
        BC      GE,    BACK         114.  + 07 1 105
```

Practical No. – 5

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

Program:-

```
#include<stdio.h>
#include<conio.h>
#include<string.h>
int main()
{

        char name[20];
        int i,l;
        printf("\n Enter a string  = ");
        gets(name);
        l = strlen(name);
        for(i=0;i<l;i++)
        {
                if(name[i]>=65 && name[i]<=90)
                {
                        printf("\n The character %c is in upper case",name[i]);
                }
                else if(name[i]>=97 && name[i]<=122)
                {
                        printf("\n The character %c is in lower case",name[i]);
                }
                else if(name[i]>=48 && name[i]<=57)
                {
                        printf("\n The character %c is a number ",name[i]);
```

```
}
else if(name[i]==' ')
{
        name[i]++;
        printf("\n the character %c is space",name[i]);
}
else if(name[i]=='\t')
{
        name[i]++;
        printf("\n the character %c is tab",name[i]);
}
else
{
        if(name[i] == 42)
        {
                printf("\n It is the multiplication operator %c",name[i]);
        }
        else if(name[i] == 43)
        {
                printf("\n It is the addition operator %c",name[i]);
        }
        else if(name[i] == 45)
        {
                printf("\n It is the subtraction operator %c",name[i]);
        }
        else if(name[i] == 47)
        {
                printf("\n It is the division operator %c",name[i]);
        }
        else if (name[i] ==' ' )
```

```
                    {
                        name[i]++;
                        printf("\n it is considered as a space %c", name[i]);
                    }
                    else if(name[i]=='\t')
                    {
                        name[i]++;
                        printf("\n it is considered as a tab %c", name[i]);
                    }
                    else if(name[i]=='\n')
                    {
                        name[i]++;
                        printf("\n it is enter key %c",name[i]);
                    }
                    else
                    {
                        printf("\n Another special character");
                    }
                }
            }
            getch();

}
```

Output:

```
Enter a string  = a + b

The character a is in lower case
the character ! is space
It is the addition operator +
the character ! is space
The character b is in lower case
```

Practical No. – 6

Aim: - Write a program that generates infix to postfix string.

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:

```
Enter infix: a+b*d
abd*+
Process returned 0 (0x0)   execution time : 9.119 s
Press any key to continue.
```

Practical No. – 7

Aim: - Write a program that generates infix to prefix string.

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,j,p=0;
    char infix[100],reverse[100],prefix[100];
    printf("Enter infix: ");
    scanf("%s",reverse);
    int length = strlen(reverse);
    j = length-1;
    for(i = 0; i < length; i++)
    {
        infix[i] = reverse[j];
        j--;
    }
    for(i = 0; i < length; i++)
    {
        if(infix[i] == '(')
            infix[i] = ')';
        else if(infix[i] == ')')
            infix[i] = '(';
    }

    for(i = 0; infix[i] != '\0'; i++ )
    {
        switch(infix[i])
        {
            case '(':
                    push(infix[i]);
                    break;
```

```
case ')':
        while( stack[top] != '(')
        {
            char temp = pop();
            prefix[p] = temp;
            p++;
        }
        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();
        prefix[p] = temp;
        p++;
        //printf("%c",temp);
    }
    push(infix[i]);
}
```

```
            break;
            default:
            prefix[p] = infix[i];
            p++;
            //printf("%c",infix[i]);
            break;
        }
    }
    //pop the remaining stack
    while(top >= 0)
    {
        char temp = pop();
        prefix[p] = temp;
        p++;
        //printf("%c",temp);
    }
    char final[100];
    int lengthPrefix = strlen(prefix);
    j = lengthPrefix - 1;
    for(i = 0; i < lengthPrefix; i++)
    {
        final[i] = prefix[j];
        j--;
    }

    for(i = 0; i < lengthPrefix; i++)
    {
        if(final[i] == '(')
            final[i] = ')';
        else if(final[i] == ')')
```

```
        final[i] = '(';
    }

    for(i = 0; i < lengthPrefix; i++)
        printf("%c",final[i]);
    printf("\n");
}
```

Output: