DATA STRUCTURES LAB – MCA 205P

1. Write a program to sort the array in ascending/descending order using merge sort.

```
#include <stdio.h>
//function declaration
void merge(int *a, int m, int *b, int n, int *c);
int main(){
//variable declaration
int a[3] = {1, 4, 6}, b[4] = {2, 3, 5, 7}, c[7], i;
//merge
merge(a, 3, b, 4, c);
//output
for(i = 0; i < 7; i++)
printf("%d ", c[i]);
printf("\nMerge complete.\n");
return 0;
}
//function definition
void merge(int *a, int m, int *b, int n, int *c){
int pa = 0, pb = 0, pc = 0;
while(pa < m && pb < n){
if(a[pa] < b[pb])
c[pc++] = a[pa++];
else
c[pc++] = b[pb++];
}
if(pa == m)
while(pb < n) //Array A exhausted
c[pc++] = b[pb++];
else
while(pa < m) //Array B exhausted
c[pc++] = a[pa++];
}
```

2. Write a program to solve the problem of towers of hanoi with 3 pegs and N discs.

```
#include <stdio.h>
#include<conio.h>
void t(int n, char beg, char aux, char end);
int main(){
int n;
clrscr();
printf("Enter the number of disks\n");
scanf("%d",&n);
t(n, 'a', 'b', 'c'); //N = (no. of disks) a, b, c are the three pegs
return 0;
}//main() ends here
void t(int n,char beg,char aux,char end){
if(n == 1){
```

```
printf("%c --> %c\n", beg, end);
}else{
t(n-1, beg, end, aux);
t(1, beg, aux, end);
t(n-1, aux, beg, end);
}
}//t() ends here
```

3.Write a program to convert the given infix expression into its postfix form.

```
#include<stdio.h>
#include<conio.h>
#include<stdlib.h>
#define size 100
int stack[size] ;
int top = -1 ;
void push(int value)
{
top++ ;
stack[top] = value ;
}
int pop()
{
int a ;
a = stack[top] ;
top-- ;
return a ;
}
int is_operand(char ch)
{
if(ch >= 'a' && ch <= 'z' || ch >='A' && ch <= 'Z')
return 1 ;
else
return 0 ;
}
int main()
{
char postfix[size] , ch ;
int i = 0 , op1 , op2 , result , m ;
clrscr();
printf("\nENTER THE POSTFIX EXPRESSION :\n");
gets(postfix) ;
while(postfix[i] != '\0')
{
ch = postfix[i] ;
if(is_operand(ch) == 1)
{
printf("ENTER THE VALUE OF %c => ",ch) ;
scanf("%d",&m) ;
push(m) ;
```

```
}
else
{
op2 = pop() ;
op1 = pop() ;
switch(ch)
{
case '+' :
result = op1 + op2 ;
push(result) ;
break ;
case '-' :
result = op1 - op2 ;
push(result) ;
break ;
case '*' :
result = op1 * op2 ;
push(result) ;
break ;
}
}
i++ ;
}
result = pop() ;
printf("\nthe result is: %d",result) ;
getch() ;
return 0 ;
```