Practical:1

AIM: 1A) Declare a calendar as an array of 7 elements (A dynamically Created array) to represent 7 days of a week. Each Element of the array is a structure having three fields. The first field is the name of the Day (A dynamically allocated String), The second field is the date of the Day (A integer), the third field is the description of the activity for a particular day (A dynamically allocated String).

PROGRAM CODE:

```
#include <stdio.h>

#include <string.h>

// Define a structure to represent a day

struct Day {

        char name[20];

        int date;

        char activity[100];

};

int main() {

 // Declare an array of 7 elements to represent the calendar

struct Day calendar[7];

// Initialize the calendar with sample data

strcpy(calendar[0].name, "Monday");

calendar[0].date = 1;

strcpy(calendar[0].activity, "Work from 9 AM to 5 PM");

strcpy(calendar[1].name, "Tuesday");


calendar[1].date = 2;

strcpy(calendar[1].activity, "Meeting at 10 AM");

strcpy(calendar[2].name, "Wednesday");

calendar[2].date = 3;

strcpy(calendar[2].activity, "Gym at 6 PM");

strcpy(calendar[3].name, "Thursday");
```

```
calendar[3].date = 4;
strcpy(calendar[3].activity, "Dinner with friends at 7 PM");
strcpy(calendar[4].name, "Friday");
calendar[4].date = 5;
strcpy(calendar[4].activity, "Movie night at 8 PM");
strcpy(calendar[5].name, "Saturday");
calendar[5].date = 6;
strcpy(calendar[5].activity, "Weekend getaway");
strcpy(calendar[6].name, "Sunday");
calendar[6].date = 7;
strcpy(calendar[6].activity, "Relax and recharge");
 // Print the calendar    printf("Calendar for the week:\n");
for (int i = 0; i < 7; i++) {
printf("%s (Date: %d): %s\n", calendar[i].name, calendar[i].date, calendar[i].activity);
}
        return 0;
}
```

OUTPUT:

```
Calendar for the week:
Monday (Date: 1): Work from 9 AM to 5 PM
Tuesday (Date: 2): Meeting at 10 AM
Wednesday (Date: 3): Gym at 6 PM
Thursday (Date: 4): Dinner with friends at 7 PM
Friday (Date: 5): Movie night at 8 PM
Saturday (Date: 6): Weekend getaway
Sunday (Date: 7): Relax and recharge
```

AIM: 1B) Write functions create(), read() and display(); to create the calendar, to read the data from the keyboard and to print weeks activity details report on screen.

PROGRAM CODE:

```
#include <stdio.h>
#include <string.h>
// Define a structure to represent a day
struct Day {
        char name[20];
        int date;
        char activity[100];
};
// Function to create the calendar
void create(struct Day calendar[7]) {
for (int i = 0; i < 7; i++) {
        printf("Enter details for %s:\n", calendar[i].name);
        printf("Date: ");
        scanf("%d", &calendar[i].date);
        printf("Activity: ");
        scanf(" %[^\n]", calendar[i].activity);
        }
}
// Function to read data from the keyboard
void read(struct Day calendar[7]) {
FILE *file = fopen("calendar.txt", "r");
if (file == NULL) {
printf("Error opening the file.\n");
return;
```

```
}
for (int i = 0; i < 7; i++) {
        fscanf(file, "%d", &calendar[i].date);
        fscanf(file, " %[^\n]", calendar[i].activity);
 }
fclose(file);
}
// Function to display the calendar
void display(struct Day calendar[7]) {
printf("Calendar for the week:\n");
for (int i = 0; i < 7; i++) {
printf("%s (Date: %d): %s\n", calendar[i].name, calendar[i].date, calendar[i].activity);
  }
}
int main() {
        struct Day calendar[7];
        // Initialize the names of the days
        strcpy(calendar[0].name, "Monday");
        strcpy(calendar[1].name, "Tuesday");
        strcpy(calendar[2].name, "Wednesday");
        strcpy(calendar[3].name, "Thursday");
        strcpy(calendar[4].name, "Friday");
        strcpy(calendar[5].name, "Saturday");
        strcpy(calendar[6].name, "Sunday");

        int choice;
        printf("1. Create Calendar\n");
        printf("2. Read Calendar from File\n");
```

```
printf("Enter your choice: ");
scanf("%d", &choice);
switch (choice) {
case 1:     create(calendar);
            break;
case 2:     read(calendar);
            break;
default:
        printf("Invalid choice.\n");
        return 1;
}
display(calendar);
return 0;
}
```

OUTPUT:

```
1. Create Calendar
2. Read Calendar from File
Enter your choice: 1
Enter details for Monday:
Date: 10
Activity: Work from home
Enter details for Tuesday:
Date: 11
Activity: Work 9am to 5pm from office
Enter details for Wednesday:
Date: 12
Activity: Outdoor activity
Enter details for Thursday:
Date: 12
Activity: Office work
Enter details for Friday:
Date: 13
Activity: Office wotk
Enter details for Saturday:
Date: 14
Activity: Meeting at 10 AM
Enter details for Sunday:
Date: Relax and Reachrge
Activity: Calendar for the week:
Monday (Date: 10): Work from home
Tuesday (Date: 11): Work 9am to 5pm from office
Wednesday (Date: 12): Outdoor activity
Thursday (Date: 12): Office work
Friday (Date: 13): Office wotk
Saturday (Date: 14): Meeting at 10 AM
Sunday (Date: 1995719452): Relax and Reachrge
```

Practical:2

AIM: Develop a Program in C for the following operations on Strings.
a. Read a main String (STR), a Pattern String (PAT) and a Replace String (REP)
b. Perform Pattern Matching Operation: Find and Replace all occurrences of PAT in STR with REP if PAT exists in STR. Report suitable messages in case PAT does not exist in STR
Support the program with functions for each of the above operations. Don't use Built-in functions.

PROGRAM CODE:

```
#include<stdio.h>
#include<conio.h>
void main()
{
char STR[100],PAT[100],REP[100],ans[100];
int i,j,c,m,k,flag=0;
printf("\nEnter the MAIN string: \n"); gets(STR);
printf("\nEnter a PATTERN string: \n"); gets(PAT);
printf("\nEnter a REPLACE string: \n"); gets(REP);
i = m = c = j = 0;
while ( STR[c] != '\0') {
// Checking for Match
if ( STR[m] == PAT[i]) {
i++; m++;
flag=1;
if ( PAT[i] == '\0')
{
//copy replace string in ans string
for(k=0; REP[k] != '\0';k++,j++)
ans[j] = REP[k];
i=0;
c=m;
```

```
} }
else //mismatch
{
ans[j] = STR[c];
j++; c++;
m = c; i=0;
} }
if(flag==0)
{
printf("Pattern doesn't found!!!");
}
else
{
ans[j] = '\0';
printf("\nThe RESULTANT string is:%s\n" ,ans);
}
getch();
}
```

OUTPUT:

```
Enter the MAIN string:
welcome

Enter a PATTERN string:
come

Enter a REPLACE string:
done

The RESULTANT string is:weldone
```

```
Enter the MAIN string:
studio

Enter a PATTERN string:
queen

Enter a REPLACE string:
name
Pattern doesn't found!!!
```

Practical:3

AIM: Develop a menu driven Program in C for the following operations on STACK of Integers (Array Implementation of Stack with maximum size MAX)
a. Push an Element on to Stack
b. Pop an Element from Stack
c. Demonstrate how Stack can be used to check Palindrome
d. Demonstrate Overflow and Underflow situations on Stack
e. Display the status of Stack
f. Exit
Support the program with appropriate functions for each of the above operations.

PROGRAM CODE:

```
#include<stdlib.h>

#include<stdio.h>

#include<string.h>

#define max_size 5

int stack[max_size],top=-1,flag=1;

int i,temp,item,rev[max_size],num[max_size];

void push();

void pop();

void display();

void pali();

void main()

{

int choice;

printf("\n\n--------STACK OPERATIONS--------");

printf("1.Push\n");

printf("2.Pop\n"); printf("3.Palindrome\n"); printf("4.Display\n"); printf("5.Exit\n");

printf(" ");

while(1)

{
```

```
printf("\nEnter your choice:\t");
scanf("%d",&choice);
switch(choice)
{
case 1: push();break;
case 2: pop();
if(flag)
printf("\nThe poped element: %d\t",item);
temp=top; break;
case 3: pali();
top=temp; break;
case 4: display(); break;
case 5: exit(0); break;
default: printf("\nInvalid choice:\n"); break;
} }
//return 0;
getch();
}

void push() //Inserting element into the stack
{
if(top==(max_size-1))
{
printf("\nStack Overflow:");
}
else
{
printf("Enter the element to be inserted:\t");
```

```
scanf("%d",&item);
top=top+1;
stack[top]=item;
}
temp=top;
}
void pop() //deleting an element from the stack
{
if(top==-1)
{
printf("Stack Underflow:");
flag=0;
}
else
{
item=stack[top];
top=top-1;
}
}

void pali()
{ i=0;
if(top==-1)
{
printf("Push some elements into the stack first\n");
}
else
{
```

```
while(top!=-1)
{
rev[top]=stack[top]; pop();
}
top=temp; for(i=0;i<=temp;i++)
{
if(stack[top--]==rev[i])
{
if(i==temp)
{
printf("Palindrome\n"); return;
}
}
}
}
printf("Not Palindrome\n");
}
void display()
{
int i; top=temp;
if(top==-1)
{
printf("\nStack is Empty:");
}
else
{
printf("\nThe stack elements are:\n" );
for(i=top;i>=0;i--)
```

```
{
printf("%d\n",stack[i]);
}
}
}
```

OUTPUT:

```
--------STACK OPERATIONS--------1.Push
2.Pop
3.Palindrome
4.Display
5.Exit

Enter your choice:      1
Enter the element to be inserted:       2

Enter your choice:      2

The poped element: 2
Enter your choice:      4

Stack is Empty:
Enter your choice:      1
Enter the element to be inserted:       2

Enter your choice:      1
Enter the element to be inserted:       3

Enter your choice:      1
Enter the element to be inserted:       4

Enter your choice:      4

The stack elements are:
4
3
2

Enter your choice:
```

```
Enter your choice:      1
Enter the element to be inserted:       5

Enter your choice:      1
Enter the element to be inserted:       6

Enter your choice:      1

Stack Overflow:
Enter your choice:      4

The stack elements are:
6
5
4
3
2

Enter your choice:      3
Not Palindrome

Enter your choice:      1

Stack Overflow:
Enter your choice:
```

Practical:4

AIM: Develop a Program in C for converting an Infix Expression to Postfix Expression. Program should support for both parenthesized and free parenthesized
expressions with the operators: +, -, *, /, % (Remainder), ^ (Power) and alphanumeric operands.

PROGRAM CODE:

```
#define SIZE 50 /* Size of Stack */

#include <ctype.h>

#include <stdio.h>

char s[SIZE];

int top = -1; /* Global declarations */

push(char elem) /* Function for PUSH operation */

{

s[++top] = elem;

}

char pop() /* Function for POP operation */

{

return (s[top--]);

}

int pr(char elem) /* Function for precedence */

{

switch (elem)

{

case '#': return 0;

case '(': return 1;

case '+':

case '-': return 2;

case '*': case '/':

case '%': return 3;
```

```
case '^': return 4;
} }
void main() /* Main Program */
{
char infx[50], pofx[50], ch, elem;
int i = 0, k = 0;
printf("\n\nRead the Infix Expression ? ");
scanf("%s", infx);
push('#');
while ((ch = infx[i++]) != '\0')
{
if (ch == '(') push(ch);
else if (isalnum(ch))
pofx[k++] = ch;
else if (ch == ')')
{
while (s[top] != '(')
pofx[k++] = pop();
elem = pop(); /* Remove ( */
}
else /* Operator */
{
while (pr(s[top]) >= pr(ch))
pofx[k++] = pop();
push(ch);
} }
while (s[top] != '#') /* Pop from stack till empty */
pofx[k++] = pop();
```

```
pofx[k] = '\0'; /* Make pofx as valid string */
printf("\n\nGiven Infix Expn: %s Postfix Expn: %s\n", infx, pofx);
}
```

OUTPUT:

```
Read the Infix Expression ? (a+b)-(c^d)/e

Given Infix Expn: (a+b)-(c^d)/e Postfix Expn: ab+cd^e/-
```

Practical:5

AIM: Develop a Program in C for the following Stack Applications
a. Evaluation of Suffix expression with single digit operands and operators: +, -, *, /, %, ^
b. Solving Tower of Hanoi problem with n disks

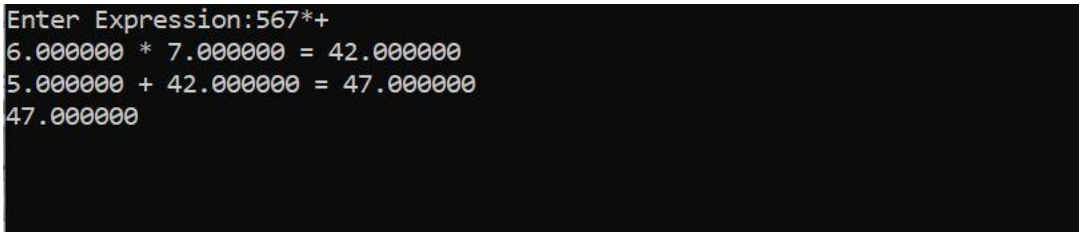
PROGRAM CODE:

```
//Evaluation of Suffix expression
#include <stdio.h>
#include <conio.h>
#include <math.h>
#define MAX 20
struct stack
{
int top;
float str[MAX];
}s;//stack
char postfix[MAX];//postfix
void push(float);
float pop();
int isoperand(char);
float operate(float,float,char);
int main()
{
int i=0;
printf("Enter Expression:");
scanf("%s",postfix);
float ans,op1,op2;
while(postfix[i]!='\0')
{
```

```
if(isoperand(postfix[i]))
push(postfix[i]-48);
else
{
op1=pop();
op2=pop();
ans=operate(op1,op2,postfix[i]);
push(ans);
printf("%f %c %f = %f\n",op2,postfix[i],op1,ans);
}
i++;
}
printf("%f",s.str[s.top]);
getch();
}
int isoperand(char x)
{
if(x>='0' && x<='9')
return 1;
else return 0;
}
void push(float x)
{
if(s.top==MAX-1)
printf("Stack is full\nStack overflow\n");
else
{
s.top++;
```

```
s.str[s.top]=x;
} }
float pop()
{
if(s.top==-1)
{
printf("Stack is emplty\nSTACK UNDERFLOW\n");
getch();
}
else
{
s.top--;
return s.str[s.top+1];
} }
float operate(float op1,float op2,char a)
{
switch(a)
{
case '+' : return op2+op1;
case '-' : return op2-op1;
case '*' : return op2*op1;
case '/' : return op2/op1;
case '^' : return pow(op2,op1);
} }
```

OUTPUT:



PROGRAM CODE:

```
//5B Solving Tower of Hanoi problem with n disks
#include <stdio.h>
#include <conio.h>
void tower(int n, int source, int temp,int destination)
{
if(n == 0)
return;
tower(n-1, source, destination, temp);
printf("\nMove disc %d from %c to %c", n, source, destination);
tower(n-1, temp, source, destination);
}
int main()
{
int n;
printf("\nEnter the number of discs: \n");
scanf("%d", &n);
tower(n, 'A', 'B', 'C');
printf("\n\nTotal Number of moves are: %d", (int)pow(2,n)-1);
return 0;
}
```

OUTPUT:

```
Enter the number of discs:
2

Move disc 1 from A to B
Move disc 2 from A to C
Move disc 1 from B to C

Total Number of moves are: 3
```

```
Enter the number of discs:
4

Move disc 1 from A to B
Move disc 2 from A to C
Move disc 1 from B to C
Move disc 3 from A to B
Move disc 1 from C to A
Move disc 2 from C to B
Move disc 1 from A to B
Move disc 4 from A to C
Move disc 1 from B to C
Move disc 2 from B to A
Move disc 1 from C to A
Move disc 3 from B to C
Move disc 1 from A to B
Move disc 2 from A to C
Move disc 1 from B to C

Total Number of moves are: 15
```