1. Develop a Program in C for the following:
a) 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).
b) 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.

```
#include<stdio.h>
#include<stdlib.h>
#include<string.h>
typedef struct
{
        char *day;
        int date;
        char *activity;
} calendar;

calendar* create()
{
        calendar *week;
        week=(calendar*)calloc(7,sizeof(calendar));
        return week;
}

void read(calendar *week)
{
        int i;
        char day[20],activity[50];
        printf("Enter week details(week day,date, activity)");
        for(i=0;i<7;i++)
        {
                printf("Day %d:",i+1);
                scanf("%s%d%s",day,&week[i].date,activity);
                week[i].day = strdup(day);
                week[i].activity = strdup(activity);
        }
}
void display(calendar *week)
{
        int i;
        printf("Week activity\nDay\tDate\tActivity\n");
        for(i=0;i<7;i++)
        {
                printf("%s\t%d\t%s\n",week[i].day,week[i].date,week[i].activity);
        }
}
int main()
{
        int choice;
        calendar *week;
        while(1)
        {
                printf("1.Create 2.Read  3.Display  4.Exit  choice:");
                scanf("%d",&choice);
                switch(choice)
                {
                        case 1: week = create();
                                        if(week!=NULL)
                                                printf("Calander is successfully created\n");
                                        break;
                        case 2: read(week);
                                        break;
                        case 3: display(week);
                                        break;
                        case 4: return 0;
                        default: printf("Invalid choice\n");
                }
        }
}
```

Sample input and output:

```
1.Create 2.Read  3.Display  4.Exit  choice:1
Calander is successfully created
1.Create 2.Read  3.Display  4.Exit  choice:2
Enter week details(week day,date, activity)Day 1:Monday
11
College
Day 2:Tuesday
12
Assignment
Day 3:Wednesday
13
Projects
Day 4:Thursday
14
Coding
Day 5:Friday
15
Dance
Day 6:Saturday
16
Music
Day 7:Sunday
17
Relax
1.Create 2.Read  3.Display  4.Exit  choice:3
```

```
Week activity
Day     Date    Activity
Monday  11      College
Tuesday 12      Assignment
Wednesday       13      Projects
Thursday        14      Coding
Friday  15      Dance
Saturday        16      Music
Sunday  17      Relax
1.Create 2.Read  3.Display  4.Exit  choice:4

...Program finished with exit code 0
Press ENTER to exit console.
```

2. Design, Develop and Implement a Program in C for the following operations on Strings

Read a main String (STR), a Pattern String (PAT) and a Replace String (REP) .
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.

```
#include<stdio.h>
char str[100],pat[100],rep[100],ans[100];
void read()
{
        printf("enter the string: ");
        gets(str);
        printf(" \n enter the patter string: ");
        gets(pat);
        printf("\n enter the replacement string: ");
        gets(rep);
}
void pat_match()
{
        int i,j,c,m,k;
        int flag=0;
        i=m=c=j=0;
        while(str[c]!='\0')
        {
                if(str[m]==pat[i])//Pattern matching
                {
                        i++;
                        m++;
                        if(pat[i]=='\0')
                        {
                                printf("\n pat:%s is found at position %d",pat,c);
                                for(k=0;rep[k]!='\0';k++,j++)
                                        ans[j]=rep[k];
                                i=0;
                                c=m;
                                flag=1;
                        }
                }
                else//pattern mismatch
                {
                        ans[j]=str[c];
                        j++;
                        c++;
                        m=c;
                        i=0;
                }
        }
        ans[j]='\0';
        if(flag==0)
                printf("\n PAT:%s is not found in STR:%s",pat,str);
        else
                printf("\n The resulting string is: %s",ans);
}
void main()
{

        read();
        pat_match();

}
```

OUTPUT:-

```
enter the string: hi rns hi
 enter the pattern string: hi
 enter the replacement string: hello
pat:hi is found at position 0
pat:hi is found at position 7
 The resulting string is: hello rns hello

enter the string: hi rns hi
 enter the pattern string: rnsit
 enter the replacement string: sjbit
PAT: rnsit is not found in STR: hi rns hi
```

3. 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

```
#include<stdio.h>
#include<math.h>
#define max 5
int stack_list[max],top=-1;
void push(int m)
{
        if(top==max-1)
                printf("\n Stack overflow");
        else
        {
                top++;
                stack_list[top]=m;
        }
}
int pop()
{
        if(top==-1)
        {
                printf("\n Stack underflow");
                return -1;
        }
        else
                return stack_list[top--];
}
void display()
{
        int i;
        if(top==-1)
                printf("\n Stack is empty");
        else
        {
                printf("\n The elements are\n");
                for(i=top;i>=0;i--)
                        printf("%d\n",stack_list[i]);
        }
}
void palindrome()
{
        int n,num,rem,i;
        printf("\n Enter n");
        scanf("%d",&n);
        top=-1;
        num=n;
        while(num!=0)
        {
                rem=num%10;
                push(rem);
                num=num/10;
        }
        num=0;
        for(i=0;top!=-1;i++)
                num=pop()*pow(10,i)+num;

        if(n==num)
                printf("\n It is a palindrome");
        else
                printf("\n It is not a palindrome");
}
int main()
{
        int c,m;
        while(1)
        {       printf("\n Enter 1-push\n2-pop\n3-display\n4-palindrome");
                scanf("%d",&c);
                switch(c)
                {
                        case 1:  printf("\n Enter an element\t");
                                scanf("%d",&m);
                                push(m);
                                break;
                        case 2:m=pop();
                                if(m!=-1)
                                         printf("\n The popped elementis %d",m);
                                break;
                        case 3:  display();
                                        break;
                        case 4:  palindrome();
                                        break;
                        default:return 0;
                }
        }
}
```

Sample Input & Output:

```
 Enter your choice
1-push
2-pop
```

```
3-display
4-palindrome
1

 Enter an element to be pushed  10

 Enter your choice
1-push
2-pop
3-display
4-palindrome1

 Enter an element to be pushed  20

 Enter your choice
1-push
2-pop
3-display
4-palindrome1

 Enter an element to be pushed  30

 Enter your choice
1-push
2-pop
3-display
4-palindrome
1

 Enter an element to be pushed  40

 Stack overflow
 Enter your choice
1-push
2-pop
3-display
4-palindrome
3

 The elements are
30
20
10

 Enter your choice
1-push
2-pop
3-display
```

```
4-palindrome
2

 The popped element is 30
 Enter your choice
1-push
2-pop
3-display
4-palindrome
2

 The popped element is 20
 Enter your choice
1-push
2-pop
3-display
4-palindrome2

 The popped element is 10
 Enter your choice
1-push
2-pop
3-display
4-palindrome
2

 Stack underflow


 Enter your choice
1-push
2-pop
3-display
4-palindrome
4

 Enter n
123

 It is not a palindrome
 Enter your choice
1-push
2-pop
3-display
4-palindrome
4

 Enter n
```

```
121

 It is a palindrome
 Enter your choice
1-push
2-pop
3-display
4-palindrome
```

4. Design, Develop and Implement 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.

```
#include<stdio.h>
#include<conio.h>
#include<ctype.h>
char stk[100];
int tos = -1;
void push(char opr)
{
        stk[++tos] = opr;
}
char pop()
{
        return(stk[tos--]);
}
int preced(char opr)
{
        if(opr=='^'||opr=='%') return(4);
        if(opr=='*'||opr=='/') return(3);
        if(opr=='+'||opr=='-') return(2);
        if(opr=='('||opr=='#') return(1);
}
void main()
{
        char infix[20],postfix[20];
        int i,j=0;
        printf("\nEnter valid INFIX expression\n");
        scanf("%s",infix);
        push('#');
        for(i=0; infix[i]!='\0'; i++)
        {
                if(infix[i]=='(')
                push('(');
                else if(isalnum(infix[i]))
                        postfix[j++] = infix[i];
                else if(infix[i]==')')
                {
                         while(stk[tos] != '(')
                              postfix[j++] = pop();
                        pop();
                }
    else
                {
                        while(preced(stk[tos]) >= preced(infix[i]))
                            postfix[j++] = pop();
                         push(infix[i]);
```

```
                                }
}
while(stk[tos] != '#')
            postfix[j++] = pop();
postfix[j]='\0';
printf("\n INFIX EXPRESSION = %s",infix);
printf("\n POSTFIX EXPRESSION = %s",postfix);

}
```

OUTPUT:-

```
Enter valid INFIX expression
(a+b*c)-d^e/f%g
INFIX EXPRESSION = (a+b*c)-d^e/f%g
POSTFIX EXPRESSION = abc*+de^fg%/-

Enter valid INFIX expression
(1+3*6)-5%6^8
INFIX EXPRESSION = (1+3*6)-5%6^8
POSTFIX EXPRESSION = 136*+56%8^-
```

5 Design, Develop and Implement a program in C for the following stack operations

a. Evaluation of suffix expression with single digit operands and operators:+,-,*,/,%,^.

b. Solving tower of Hanoi problem with n disks.

a)

```
#include<stdio.h>
#include<stdlib.h>
#include<string.h>
#include<conio.h>
int stk[25],tos=-1;
void push(int item)
{
 stk[++tos]=item;
}
int pop()
{
 return (stk[tos--]);
}
int main()
{
 char post[25],sym;
 int op1,op2,i;
printf("Enter the postfix expression:\n");
 scanf("%s",post);
 for(i=0;i<strlen(post);i++)
 {
  sym=post[i];
  switch(sym)
  {
   case '+':op2=pop();
            op1=pop();
            push(op1+op2);
            break;
   case '-':op2=pop();
            op1=pop();
            push(op1-op2);
            break;
   case '*':op2=pop();
            op1=pop();
            push(op1*op2);
            break;
   case '/':op2=pop();
            op1=pop();
            push(op1/op2);
            break;
default:push(sym-'0');
           break;
  }
 }
printf("The result if %d\n",pop());
}
```

Output1:

```
Enter the postfix expression:
6523+8*+3+*
The result if 288
```

Output2:

```
Enter the postfix expression:
93*1-2+4*5-
The result if 107
```

b)

```
#include<stdio.h>
void tower(int num,char src,char tmp,char dest)
{
 if(num==1)
 {
printf("Move disk 1 from peg %c to peg %c\n",src,dest);
  return;
 }
 tower(num-1,src,dest,tmp);
printf("Move disk %d from peg %c to peg %c\n",num,src,dest);
 tower(num-1,tmp,src,dest);
}
int main()
{
 int num;
printf("Enter number of disks\n");
 scanf("%d",&num);
 tower(num,'A','B','C');
}
```

Output1:

```
Enter number of disks
3
Move disk 1 from peg A to peg C
Move disk 2 from peg A to peg B
Move disk 1 from peg C to peg B
Move disk 3 from peg A to peg C
Move disk 1 from peg B to peg A
Move disk 2 from peg B to peg C
Move disk 1 from peg A to peg C
```