2. 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.

→
```c
#include <stdio.h>

void main(){
        char STR[100],PAT[100],REP[100],ANS[100];
        int c,i,j,m,k,flag;
        printf("Enter the MAIN string : ");
        scanf("%[^\n]",STR);
        printf("Enter the PATTERN string : ");
        scanf("%s",PAT);
        printf("Enter the REPLACE string : ");
        scanf("%s",REP);
        c=i=j=m=k=flag=0;

        while(STR[c]!='\0'){
                if(STR[m]==PAT[i]){
                        i++;
                        m++;
                        flag=1;
                        if(PAT[i]=='\0'){
                                for(k=0;REP[k]!='\0';j++,k++)
                                ANS[j]=REP[k];
                                        i=0;
                                        c=m;
                        }
                }else{
                        ANS[j]=STR[c];
                        j++;
                        c++;
                        m=c;
                        i=0;
                }
        }
        if(flag==0)
                printf("Pattern doesn't found !\n");
        else{
                ANS[j]='\0';
                printf("\nThe resultant string is %s\n",ANS);
        }
}
```

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.

→
```c
#include<stdio.h>
#include<string.h>
#include<stdlib.h>

#define max_size 5

int stack[max_size],top=-1,flag=1;
int i,ispali,temp,item,rev[max_size],num[max_size];

void push(){
	if(top==(max_size-1))
		printf("\nStack Overflow !");
	else{
		printf("Enter the element to be inserted : ");
		scanf("%d",&item);
		top++;
		stack[top]=item;
	}
	temp=top;
}

void pop(){
	if(top==-1){
		printf("Stack Underflow !\n");
		flag=0;
	}else{
		item=stack[top];
		top--;
	}
}

void pali(){
	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]&&i==temp)
				ispali=1;
			else
				ispali=0;
	}
		if(ispali)
			printf("Palindrome !\n");
		else
			printf("Not Palindrome\n");
}

void display(){
	int i;
	top=temp;

	if(top==-1)
		printf("Stack is Empty !\n");
	else{
		printf("The stack elements are : ");
		for(i=top;i>=0;i--)
			printf("%d ",stack[i]);
		printf("\n");
	}
}

int main(){
	int choice;
	printf("\n\n--------STACK OPERATIONS--------\n");
	printf("1.Push\n2.Pop\n3.Palindrome\n4.Display\n5.Exit\n");
	while(1){
		printf("> ");
		scanf("%d",&choice);
		switch(choice){
			case 1:
				push();
				break;
			case 2:
				pop();
				if(flag)
					printf("\nThe poped element : %d\n",item);
				temp=top;
				break;
			case 3:
				pali();
				top=temp;
				break;
			case 4:
				display();
				break;
			case 5:
				exit(0);
				break;

			default:
				printf("\nInvalid choice !");
				break;
		}
	}
}
```

4. 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 : `+ - * / % ^` and alphanumeric operands

→
```
#include <ctype.h>
#include <stdio.h>

#define SIZE 50

char s[SIZE];
int top=-1;

void push(char elem){
        s[++top]=elem;
}

char pop(){
        return (s[top--]);
}

int pr(char elem){
        switch(elem){
                case '#':
                        return 0;
                case '(':
                        return 1;
                case '+':
                case '-':
                        return 2;
                case '*':
                case '/':
                case '%':
                        return 3;
                case '^':
                        return 4;
        }
}

void main(){
        char infx[50],pofx[50],ch,elem;
        int i=0,k=0;

        printf("\nRead the Infix Expression\n> ");
        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();
                }else{
                        while(pr(s[top])>= pr(ch))
                                pofx[k++]=pop();
                        push(ch);
                }
        }
        while(s[top]!='#')
                pofx[k++]=pop();
        pofx[k]='\0';
        printf("\nGiven Infix expression : %s",infx);
        printf("\nPostfix expression     : %s\n",pofx);
}
```

5. 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.

→ a.

```
#include<stdio.h>
#include<ctype.h>
#include<math.h>
#include<string.h>

float compute(char symbol,float op1,float op2){
        switch(symbol){
                case '+' :
                        return op1+op2;
                        break;
                case '-' :
                        return op1-op2;
                        break;
                case '*' :
                        return op1*op2;
                        break;
                case '/' :
                        return op1/op2;
                        break;
                case '$' :
                        return op1+op2;
                        break;
                default :
                        return 0;
        }
}

void main(){
        float s[20],res,op1,op2;
        int top,i;
        char postfix[20],symbol;

        printf("Enter the postfix expression:\n> ");
        scanf("%s",postfix);
        top=-1;

        for(i=0;i<strlen(postfix);i++){
                symbol=postfix[i];
                if(isdigit(symbol))
                        s[++top]=symbol-'0';
                else{
                        op2=s[top--];
                        op1=s[top--];
                        res=compute(symbol,op1,op2);
                        s[++top]=res;
                }
        }

        res=s[top--];
        printf("The result is %f\n",res);
}
```

b.

```
#include<stdio.h>
#include<math.h>

void tower(int n,char beg,char aux,char end){
        if(n==0)
                printf("Illegal, Try with non-zero Positive Integers !\n");
        else if(n==1)
                printf("Move Disc from %c to %c\n",beg,end);
        else{
                tower(n-1,beg,end,aux);
                tower(1,beg,aux,end);
                tower(n-1,aux,beg,end);
        }
}

void main(){
        int n;
        char a,b,c;
        printf("Enter the number of Discs :\n> ");
        scanf("%d",&n);
        printf("Tower of Hanoi for %d Disc has the following steps :\n",n);
        tower(n,'a','b','c');
        printf("\n\nTotal Number of moves are : %d\n",(int)pow(2,n)-1);
}
```

6. Develop a menu driven Program in C for the following operations on Circular QUEUE of Characters (Array Implementation of Queue with maximum size MAX)

   a. Insert an Element on to Circular QUEUE
   b. Delete an Element from Circular QUEUE
   c. Demonstrate Overflow and Underflow situations on Circular QUEUE
   d. Display the status of Circular QUEUE
   e. Exit

   Support the program with appropriate functions for each of the above operations

→
```
#include<stdio.h>
#include<stdlib.h>

#define max 10

int q[10],front=0,rear= -1;
void main(){
        int ch;
        void insert();
        void delete();
        void display();
        printf("\nCircular Queue operations\n");
        printf("1. Insert\n2. Delete\n3. Display\n4. Exit\n");

        while(1){
                printf("\nEnter your choice : ");
                scanf("%d",&ch);

                switch(ch){
                        case 1:
                                insert();
                                break;
                        case 2:
                                delete();
                                break;
                        case 3:
                                display();
                                break;
                        case 4:
                                exit(1);
                                default:
                        printf("Invalid option !\n");
                }
        }
}

void insert(){
        int x;
        if((front==0&&rear==max-1)||(front>0&&rear==front-1))
                printf("Queue is overflow !\n");
        else{
                printf("Enter element to be insert : ");
                scanf("%d",&x);
                if(rear==max-1&&front>0){
                        rear=0;
                        q[rear]=x;
                }else if((front==0&&rear==-1)||(rear!=front-1))
                                q[++rear]=x;
        }
}

void delete(){
        int a;
        if((front==0)&&(rear==-1)){
                printf("Queue is underflow !\n");
                exit(1);
        }
        if(front==rear){
                a=q[front];
                rear=-1;
                front=0;
        }else if(front==max-1){
                a=q[front];
                front=0;
        }else
                a=q[front++];

        printf("Deleted element is : %d\n",a);
}

void display(){
        int i,j;
        if(front==0&&rear==-1){
                printf("Queue is underflow !\n");
                exit(1);
        }
        if(front>rear){
                for(i=0;i<=rear;i++)
                        printf("%d ",q[i]);
                for(j=front;j<=max-1;j++)
                        printf("%d ",q[j]);

                printf("\nRear is at %d\n",q[rear]);
                printf("Front is at %d\n",q[front]);
        }else{
                for(i=front;i<=rear;i++)
                        printf("%d ",q[i]);

                printf("\nRear is at %d\n",q[rear]);
                printf("Front is at %d\n",q[front]);
        }
        printf("\n");
}
```

7. Design, Develop and Implement a menu driven Program in C for the following operations on SLL of Student Data with the fields : USN, Name, Branch, Sem, PhNo.
   a. Create a SLL of N Students Data by using front insertion.
   b. Display the status of SLL and count the number of nodes in it
   c. Perform Insertion and Deletion at End of SLL
   d. Perform Insertion and Deletion at Front of SLL
   e. Demonstrate how this SLL can be used as STACK and QUEUE
   f. Exit

→

```c
#include<stdio.h>
#include<stdlib.h>

struct node{
        char usn[20],name[10],branch[5];
        unsigned long long int phno;
        int sem;
        struct node *link;
};

typedef struct node *NODE;
NODE temp,FIRST=NULL;

NODE getnode(){
        NODE x;
        x=(NODE)malloc(sizeof(struct node));
        x->link==NULL;
        return x;
}

void read(){
        temp=getnode();
        printf("Enter USN : ");
        scanf("%s",temp->usn);
        printf("Enter the name :\n");
        scanf("%s",temp->name);
        printf("Enter Branch : ");
        scanf("%s",temp->branch);
        printf("Enter the phone number :\n");
        scanf("%llu",&temp->phno);
        printf("Enter the semester : ");
        scanf("%d",&temp->sem);
}

void create_SSL(){
        int n,i;
        printf("Enter the number of students : ");
        scanf("%d",&n);

        for(i=1;i<=n;i++){
                printf("\nEnter the details of student %d :\n",i);
                read();
                if(FIRST==NULL)
                        FIRST=temp;
                else{
                        temp->link=FIRST;
                        FIRST=temp;
                }
        }
}

void display_count(){
        int count=1;
        temp=FIRST;
        printf("Student details :\n");

        if(FIRST==NULL)
                printf("Student detail is NULL and the count is 0\n");
        else{
                printf("\nUSN\tNAME\t\tBRANCH\t\tPHONE\t\tSEMESTER\n");
                while(temp->link!=NULL){
                        count++;
                        printf("%s\t\t%s\t\t%s\t\t%llu\t\t%d\n", \
                                        temp->usn,temp->name,temp->branch,temp->phno,temp->sem);
                        temp=temp->link;
                }

                printf("%s\t%s\t\t%s\t\t%llu\t\t%d\n",temp->usn,temp->name,temp->branch,temp->phno,temp->sem);
                printf("\nStudent count is %d\n",count);
        }
}

void insert_front(){
        printf("Enter the details of the students\n");
        read();

        if(FIRST==NULL)
                FIRST=temp;
        else{
                temp->link=FIRST;
                FIRST=temp;
        }
}

NODE insert_end(){
        NODE last=FIRST;
        printf("Enter the details of the students :\n");
        read();

        if(FIRST==NULL)
                FIRST=temp;
        else{
                while(last->link!=NULL)
                        last=last->link;
                last->link=temp;
        }
        return FIRST;
}

void delete_front(){
        temp=FIRST;
        if(FIRST==NULL)
                printf("List is empty !\n");
        else{
                printf("Deleted item is %s\n",temp->usn);
                FIRST=FIRST->link;
                free(temp);
        }
}

void delete_end(){
        NODE last=NULL;
        temp=FIRST;

        if(FIRST==NULL)
                printf("List is empty !\n");
        else if(FIRST->link==NULL){
                printf("Deleted item is %s\n",temp->usn);
                free(FIRST);
                FIRST=NULL;
        }else{
                while(temp->link!=NULL){
                        last=temp;
                        temp=temp->link;
                }

                last->link=NULL;
                printf("Printed element is %s\n",temp->usn);
                free(temp);
        }
}

void main(){
        int choice;
        while(1){
                printf("\n1.Create SSL \n2.Display SSL \n3.Insert front\n4.Insert end");
                printf("\n5.Delete front \n6.Delete End\n7.Exit\n");
                printf("\nEnter your choice : ");
                scanf("%d",&choice);
                switch(choice){
                        case 1:
                                create_SSL();
                                break;
                        case 2:
                                display_count();
                                break;
                        case 3:
                                insert_front();
                                break;
                        case 4:
                                insert_end();
                                break;
                        case 5:
                                delete_front();
                                break;
                        case 6:
                                delete_end();
                                break;
                        case 7:
                                return;
                        default:
                                printf("Invalid choice !\n");
                }
        }
}
```