1. **Design, Develop and Implement a menu driven Program in C for the following Array operations**

   a. Creating an Array of N Integer Elements

   b. Display of Array Elements with Suitable Headings

   c. Inserting an Element (ELEM) at a given valid Position (POS)

   d. Deleting an Element at a given valid Position(POS)

   e. Exit.

   Support the program with functions for each of the above operations

```
#include<stdio.h>
#include<stdlib.h>
int n,*a;
void create(n)
{
    int i;
    a=(int*)malloc(n*sizeof(int));
    if(a==NULL)
    {
            printf("array not created\n");
            exit(0);
    }
    printf("\n array created successfully");
    printf("\n enter array elements");
    for(i=0;i<n;i++)
            scanf("%d",&a[i]);
}
void display()
{
    int i;
    for(i=0;i<n;i++)
    {
            printf("%d\t",a[i]);
    }
}
void insert(ele,pos)
{
    int j=n-1;
    n++;
    a=(int*)realloc(a,n*sizeof(int));
    while(j>=pos)
    {
             a[j+1]=a[j];
            j--;
    }
    a[pos]=ele;
}
```

```
void delet(pos)
{
    int j,item;
    item=a[pos];
    printf("the deleted ele is %d\n",item);
    for(j=pos;j<n-1;j++)
    {
            a[j]=a[j+1];
    }
    n--;
    if(n==0)
            printf("\n no elements in the array\n");
}
void main()
{
    int ch,ele,pos;
    clrscr();
    while(1)
    {
            printf("\nenter a choice\n 1.create\n 2.display\n 3.insert\n 4.delete\n 5.exit\n");
            scanf("%d",&ch);
            switch(ch)
            {
                    case 1: printf("enter the size of array\n");
                            scanf("%d",&n);
                            create(n);
                            break;
                    case 2: display();
                            break;
                    case 3: printf("enter the element\n");
                            scanf("%d",&ele);
                            label: printf("enter the position (*index starts from 0)\n");
                            scanf("%d",&pos);
                            if(pos>=0 &&pos<n)
                                    insert(ele,pos);
                            else
                                    goto label;
                            break;
                    case 4: label2: printf("enter the position(*index starts from 0)\n");
                            scanf("%d",&pos);
                            if(pos>=0 && pos<n)
                                    delet(pos);
                            else
                                    goto label2;
                            break;
                    default: free(a);
```

```
                    exit(0);
            }
    }
}
```

Without using goto statement

```
#include<stdio.h>
#include<stdlib.h>
int n,*a;
void create(n)
{
    int i;
    a=(int*)malloc(n*sizeof(int));
    if(a==NULL)
    {
            printf("array not created\n");
            exit(0);
    }
    printf("\n array created successfully");
    printf("\n enter array elements");
    for(i=0;i<n;i++)
            scanf("%d",&a[i]);
}
void display()
{
    int i;
    for(i=0;i<n;i++)
    {
            printf("%d\t",a[i]);
    }
}
void insert(ele,pos)
{
    int j=n-1;
    n++;
    a=(int*)realloc(a,n*sizeof(int));
```

```
    while(j>=pos)
    {
            a[j+1]=a[j];
           j--;
    }
    a[pos]=ele;
}
void delet(pos)
{
    int j,item;
    item=a[pos];
    printf("the deleted ele is %d\n",item);
    for(j=pos;j<n-1;j++)
    {
           a[j]=a[j+1];
    }
    n--;
    if(n==0)
           printf("\n no elements in the array\n");
}
void main()
{
    int ch,ele,pos;

    while(1)
    {
           printf("\nenter a choice\n 1.create\n 2.display\n 3.insert\n 4.delete\n 5.exit\n");
           scanf("%d",&ch);
           switch(ch)
           {
                  case 1: printf("enter the size of array\n");
                         scanf("%d",&n);
                         create(n);
                         break;
                  case 2: display();
                         break;
                  case 3: printf("enter the element\n");
                         scanf("%d",&ele);
                         while(1)
                         {
                          printf("enter the position (*index starts from 0)\n");
                         scanf("%d",&pos);
                         if(pos>=0 &&pos<=n)
                         {
                                insert(ele,pos);
                         break;
```

```
                        }
                        }
                        break;


                case 4: while(1)
                {
                printf("enter the position(*index starts from 0)\n");
                        scanf("%d",&pos);
                        if(pos>=0 && pos<n)
                        {
                                delet(pos);

                        break;
                        }
                }
        break;
                default: free(a);
                        exit(0);
                }
        }
    }
```

OUTPUT

```
enter a choice
 1.create
 2.display
 3.insert
 4.delete
 5.exit
enter the size of array
3
enter array elements
2 4 6 8 10
array created successfully
 enter a choice
 1.create
 2.display
 3.insert
 4.delete
 5.exit
2
2       4       6       8       10
enter a choice
 1.create
 2.display
```

```
 3.insert
 4.delete
 5.exit
3
enter the element
3
enter the position (*index starts from 0)
0
enter a choice
 1.create
 2.display
 3.insert
 4.delete
 5.exit
2
3        2        4        6        8        10
enter a choice
 1.create
 2.display
 3.insert
 4.delete
 5.exit
4
enter the position(*index starts from 0)
3
the deleted ele is 6
enter a choice
 1.create
 2.display
 3.insert
 4.delete
 5.exit
2
3        2        4        8        10
enter a choice
 1.create
 2.display
 3.insert
 4.delete
 5.exit
5
```

CO	PO1	PO2	PO3	PO4	PO5	PO6	PO7	PO8	PO9	PO10	PO11	PO12	PSO1	PSO2	PSO3
CO1	3	3	3							3		3	3		

2. **Design, Develop and Implement 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.

```
#include<stdio.h>
#include<conio.h>
char str[100],pat[100],rep[100],ans[100];
void read()
{
        printf("enter the string: ");
        gets(str);
        printf(" \n enter the patter string: ");
        flushall();
        gets(pat);
        printf("\n enter the replacement string: ");
        flushall();
        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()
{
        clrscr();
        read();
        pat_match();
        getch();
}
```

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
```

CO	PO1	PO2	PO3	PO4	PO5	PO6	PO7	PO8	PO9	PO10	PO11	PO12	PSO1	PSO2	PSO3
CO1	3	3	3							3		3	3		

3. Design, Develop and Implement 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<stdlib.h>
#include<stdio.h>
#include<string.h>
#define SIZE 5

int stack[SIZE],tos = -1;
void push(int);
int  pop();
void display();
void pali();

int main()
{  int choice, item;
   while(choice)
   { printf("\n\n--------STACK OPERATIONS-----------\n");
     printf("1.Push 2.Pop 3.Palindrome 4.Display 5.Exit\n");
     printf("\nEnter your choice:\t");
          scanf("%d",&choice);
    switch(choice)
    {    case 1: printf("enter element for inserion");  scanf("%d",&item);
                    push(item); break;
       case 2: if( tos == -1)
                       printf("\n\nStack is Underflow");
                  else
                  {  item = pop();
                       printf("Elememt deleted = %d",item);
                    }   break;
       case 3: pali(); break;
       case 4: display(); break;
       case 5: exit(0); break;
       default: printf("\nInvalid choice:\n");
    }
 }
return 0;
```

```
}

void push(int item)
{       if (tos == SIZE -1)
            printf("\n\nStack is Overflow");
        else
           stack[++tos] = item;
}

int pop()
{       return(stack[tos--]);
}


void pali()
{   int  rev[SIZE], i=0, len = tos , flag = 0;

        while( len >= 0)
          rev[i++] = stack[len--];

        printf("reversed array is:\n");
          for(i=0; i <= tos;  i++)
                printf("%d ",rev[i]);

        printf("checking for palindrome\n");

         for(i=0; i<= tos; i++)
           { if(rev[i] != stack[i])
              flag = 1; break;
           }

        if( flag == 0 )
                printf("It is palindrome number\n");
        else
                printf("It is not a palindrome number\n");
}

void display()
{ int i;
  if( tos == -1)
      printf("\nStack is Empty:");
  else
    { printf("\nThe stack elements are:\n" );
      for(i=tos; i>=0;i--)
        printf("%d\n",stack[i]);
    }
```

}

OUTPUT

```
1. PUSH
2. POP
3. PALINDROME
4. DISPLAY
5. Exit
Enter Your Choice: 1
Enter a element to be pushed: 10
1. PUSH
2. POP
3. PALINDROME
4. DISPLAY
5. Exit
Enter Your Choice: 1
Enter a element to be pushed: 20
1. PUSH
2. POP
3. PALINDROME
4. DISPLAY
5. Exit
Enter Your Choice: 1
Enter a element to be pushed: 10
1. PUSH
2. POP
3. PALINDROME
4. DISPLAY
5. Exit
Enter Your Choice: 1
Enter a element to be pushed: 30
1. PUSH
2. POP
3. PALINDROME
4. DISPLAY
5. Exit
Enter Your Choice: 2
Popped element is 30
1. PUSH
2. POP
3. PALINDROME
4. DISPLAY
5. Exit
Enter Your Choice: 3
Numbers= 10
Numbers= 20
```

Numbers= 10
reverse operation:
reverse array:
10
20
10
check for palindrome
It is palindrome number

1. PUSH
2. POP
3. PALINDROME
4. DISPLAY
5. Exit
Enter Your Choice: 2
Popped element is 10
1. PUSH
2. POP
3. PALINDROME
4. DISPLAY
5. Exit
Enter Your Choice: 3
Numbers= 20
Numbers= 10
reverse operation:
reverse array:
10
20
check for palindrome
It is not a palindrome number

1. PUSH
2. POP
3. PALINDROME
4. DISPLAY
5. Exit
Enter Your Choice: 4
The stack contents are:
20
10

1. PUSH
2. POP
3. PALINDROME
4. DISPLAY
5. Exit

Enter Your Choice: 5

CO	PO1	PO2	PO3	PO4	PO5	PO6	PO7	PO8	PO9	PO10	PO11	PO12	PSO1	PSO2	PSO3
CO2	3	3	3	2						3		3	3	3	

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<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);
        getch();
}
```

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^-
```

CO	PO1	PO2	PO3	PO4	PO5	PO6	PO7	PO8	PO9	PO10	PO11	PO12	PSO1	PSO2	PSO3
CO4		3	3	3						3		3	2	3	

5. Design, Develop and Implement 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

```
#include<stdio.h>
#include<string.h>
#include<math.h>
int stk[25],tos = -1;

void push(int item)
{
  stk[++tos] = item;
}

int pop()
{  return( stk[tos--] );
}

void 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;
                          case'%':  op2 = pop();   op1 = pop(); push(op1%op2); break;
case'^':  op2 = pop();   op1 = pop(); push(pow(op1,op2)); break;


                  default:  push( sym - '0' ); break;
           }
          }

   printf("The result is %d\n",pop());
}
```

OUTPUT

Enter a valid suffix expression:
6523+8*+3+*
The value of the given expression is 288

b. Solving Tower of Hanoi problem with n disks

```
#include <stdio.h>
void towers(int, char, char, char);
int main()
{
   int num;
printf("Enter the number of disks : ");
scanf("%d", &num);
printf("The sequence of moves involved in the Tower of Hanoi are :\n");
towers(num, 'A', 'C', 'B');
getch();
   return 0;
}
void towers(int num, char source, char dest, char aux)
{
   if (num == 1)
   {
printf("\n Move disk 1 from peg %c to peg %c", source,dest);
      return;
   }
towers(num - 1, source, aux,dest);
printf("\n Move disk %d from peg %c to peg %c", num, source, dest);
towers(num - 1, aux, dest, source);
}
```

OUTPUT

Enter the number of disks: 3
The sequence of moves involved in the Tower of Hanoi are:
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

CO	PO1	PO2	PO3	PO4	PO5	PO6	PO7	PO8	PO9	PO10	PO11	PO12	PSO1	PSO2	PSO3
CO4		3	3	3						3		3	2	3	