EXPERIMENT – 01

Design, Develop and Implement a menu driven program in C for the following Array operations

a. Creating 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.

PROGRAM CODE:

```
#include<stdio.h>
#include<stdlib.h>
  int a[10], n, elem, i, pos;

  void create()
  {
        printf("\nEnter the size of the array elements: ");
        scanf("%d", &n);
        printf("\nEnter the elements for the array:\n");
        for(i=0; i<n; i++)
        scanf("%d", &a[i]);
  }

  void display()
  {
        int i;
        printf("\nThe array elements are:\n");
        for(i=0; i<n; i++)
        {
              printf("%d\t", a[i]);
        }
  }

  void insert()
  {
        printf("\nEnter the position for the new element: ");
        scanf("%d", &pos);
        printf("\nEnter the element to be inserted: ");
        scanf("%d", &elem);
```

```
	for(i=n-1; i>=pos; i--)
	{
		a[i+1] = a[i];
	}
	a[pos] = elem;
	n = n+1;
}

void del()
{
	printf("\nEnter the position of the element to be deleted: ");
	scanf("%d", &pos);
	elem = a[pos];
	for(i=pos; i<n-1; i++)
	{
		a[i] = a[i+1];
	}
	n = n-1;
	printf("\nThe deleted element is = %d", elem);
}

void main()
{
	int ch;
	do{
		printf("\n\n--------Menu ----------\n");
		printf("1.Create\n 2.Display\n 3.Insert\n 4.Delete\n 5.Exit\n");
		printf(" ---------------------------- ");
		printf("\nEnter your choice: ");
		scanf("%d", &ch);
		switch(ch)
		{
			case 1: create();
				break;
			case 2: display();
				break;
			case 3: insert();
				break;
			case 4: del();
				break;
			case 5: exit(0);
				break;
```

```
                default: printf("\nInvalid choice:\n");
                        break;
                }
        }while(ch!=5);
}
```

OUTPUT:

```
-------- Menu ----------
1. Create
2. Display
3. Insert
4. Delete
5. Exit
------------------------------
Enter your choice: 1

Enter the size of the array elements (max 10): 5

Enter the elements for the array:
1 2 3 4 5


-------- Menu ----------
1. Create
2. Display
3. Insert
4. Delete
5. Exit
------------------------------
Enter your choice: 2

The array elements are:
1       2       3       4       5

-------- Menu ----------
1. Create
2. Display
3. Insert
4. Delete
5. Exit
------------------------------
Enter your choice: 3

Enter the position for the new element (0 to 5): 2

Enter the element to be inserted: 10


-------- Menu ----------
1. Create
2. Display
3. Insert
4. Delete
5. Exit
------------------------------
Enter your choice: 2

The array elements are:
1       2       10      3       4       5

-------- Menu ----------
1. Create
2. Display
3. Insert
4. Delete
5. Exit
------------------------------
Enter your choice: 4

Enter the position of the element to be deleted (0 to 5): 2

The deleted element is = 10


-------- Menu ----------
1. Create
2. Display
3. Insert
4. Delete
```

EXPERIMENT – 02

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. Repost 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<stdlib.h>

char str[100], pat[50], rep[50], ans[100];
int i, j, c, m, k, flag=0;

  void stringmatch()
  {
        i = m = c = j = 0;
        while(str[c] != '\0')
        {
              if(str[m] == pat[i])
              {
                    i++;
                    m++;
                    if(pat[i] == '\0')
                    {
                          flag = 1;
                          flag = 1;
                          for(k = 0; rep[k] != '\0'; k++, j++)
                          ans[j] = rep[k];
                          i = 0;
                    }     c = m;
              }
              else
              {
                    ans[j] = str[c];
                    j++;
```

```
                    c++;
                    m = c;
                    i = 0;
                }
            }
            ans[j] = '\0';
    }

    void main()
    {
printf("\nEnter a main string \n");
gets(str);
printf("\nEnter a pattern string \n");
gets(pat);
printf("\nEnter a replace string \n");
gets(rep);
stringmatch(); if(flag
== 1)
printf("\nThe resultant string is\n %s" , ans);
else
    printf("\nPattern string NOT found\n");
    }
```

OUTPUT:

Enter a main string: Welcome to DSC Lab
Enter a pattern string: DSC
Enter a replace string: Data structures
The result string is: Welcome to Data structures Lab

EXPERIMENT - 03

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. Display the status of stack.
d. Demonstrate Overflow and Underflow situations on stack.
e. Exit.

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

PROGRAM CODE:

```
#include<stdio.h>
#include<stdlib.h> int
i,top,ch,s[3],item;
  void push(),pop(),dis();

  void main()
  {
        top=-1;
        ch=0;
        while(ch!=4)
        {
              printf("\nEnter the menu for stack operation\n");
              printf("\n1:insert\n2:delete\n3:display\n4:exit\n");
              printf("Input ur choice\n");
              scanf("%d",&ch);
              switch(ch)
              {
                    case 1: push();
                            break;
                    case 2: pop();
                            break;
                    case 3: dis();
                            break;
                    case 4: exit(0);
                            break;
                   default: printf("Invalid choice\n");
               }
        }
```

```
        return 0;
}

void push()
{
        int item;
        if(top==3-1)
        {
                printf("Stack Overflow\n");
                return;
        }
        printf("Enter an item to be pushed:");
        scanf("%d",&item);
        top+=1;
        s[top]=item;
}

void pop()
{
        if(top==-1)
        {
                printf("Stack Undeflow\n");
                return;
        }
        printf("Item poped is %d\n",s[top--]);
}

void dis()
{
        if(top==-1)
        {
                printf("Stack is empty\n");
                return;
        }
        printf("\nStack contains .....\n");
        for(i=0; i<=top; i++)
        printf("%d\t",s[i]);
}
```

OUTPUT:

OUTPUT:

```
Enter the menu for stack operation
1: Insert
2: Delete
3: Display
4: Exit
Input your choice: 1
Enter an item to be pushed: 10

Enter the menu for stack operation
1: Insert
2: Delete
3: Display
4: Exit
Input your choice: 1
Enter an item to be pushed: 20

Enter the menu for stack operation
1: Insert
2: Delete
3: Display
4: Exit
Input your choice: 3

Stack contains:
10    20

Enter the menu for stack operation
1: Insert
2: Delete
3: Display
4: Exit
Input your choice: 2
Item popped is 20

Enter the menu for stack operation
1: Insert
2: Delete
3: Display
4: Exit
Input your choice: 3

Stack contains:
10

Enter the menu for stack operation
1: Insert
2: Delete
3: Display
4: Exit
Input your choice: 4
```

EXPERIMENT – 04

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

PROGRAM CODE:

```
#include<stdio.h>
#include<string.h>
void push(char item[],int *top,char s[20][20])
{
        *top+=1;
        strcpy(s[*top],item);
}
char *pop(int *top,char s[20][20])
{
        char *item;
        item=s[*top];
        *top-=1;
         return item;
}

void pos_to_in(char postfix[], char infix[])
{
        char s[20][20],symbol,temp[2],*op1,*op2;
        int i,top;
        top=-1;
        for(i=0; i<strlen(postfix); i++)
        {
                symbol=postfix[i];
                temp[0]=symbol;
                temp[1]='\0';
                switch(symbol)
                {
                        case '+':
                        case '-':
                        case '*':
                        case '/':
                        case '$':
                                op2=pop(&top,s);
                                op1=pop(&top,s);
                                strcpy(infix,"(");
                                strcat(infix,op1);
```

```
                        strcat(infix,temp);
                        strcat(infix,op2);
                        strcat(infix,")");
                        push(infix,&top,s);
                        break;
                        default : push(temp,&top,s);
                }
        }
}

main()
{
char postfix[25],infix[25];
printf("enter ur postfix expr\n");
scanf("%s",postfix);
pos_to_in(postfix,infix);
printf("\nThe equivalent postfix is %s\n",infix);
return 0;
}
```

execution steps: mkdir filename.c
cd filename.c

gedit filename.c
gcc filename. c
./a.out

OUTPUT:
Enter your postfix expression ab+
The equivalent postfix is (a+b)

EXPERIMENT – 05

Design, Develop and Implement a Program in C for the following Stack Applications
a. Evaluation of **Postfix expression**
b. Solving **Tower of Hanoi** problem with **n** disks.

PROGRAM CODE:
PROGRAM 5A:

```
#include<stdio.h>
#include<ctype.h>
int stack[25],top=-1;
main()
{
char postfix[25];
int i=0,value[20],result;
printf("Enter a valid postfix expr\n");
scanf("%s",postfix);
while(postfix[i]!='\0')
{
  if(isalpha(postfix[i]))
  {
    printf("enter the value %c\n",postfix[i]);
    scanf("%d",&value[i]);
  }
  i++;
}
result=eval_postfix(postfix,value);
printf("The result of %s=%d\n",postfix,result);
}
int eval_postfix(char postfix[],int data[])
{
  int i=0,op1,op2,res;
  char ch;
  while(postfix[i]!='\0')
  {
      ch=postfix[i];
      if(isalpha(ch))
      push(data[i]);
      else
      {
          op2=pop();
          op1=pop();
          switch(ch)
          {
```

```
            case '+':   push(op1+op2); break;
            case '-':   push(op1-op2); break;
            case '/':   push(op1/op2); break;
            case '*':   push(op1*op2); break;
            case '$':   push(op1^op2); break;
           }
      }
  i++;
}
res=pop();
return res;
}

  int push(int num)
  {
    top+=1;
    stack[top]=num;
    return;
  }
  int pop()
  {
  int num;
  num=stack[top--];
  return num;
  }
```

OUTPUT:

```
Enter a valid postfix expr  ab+
enter the value a           10
enter the value b           5
The result of ab+ =         15
```

PROGRAM 5B:

```
#include<stdio.h>
#include<stdlib.h>
#include<math.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);
   }
   void 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);
}
```

for execution: gcc filename.c -lm

OUTPUT:

```
Enter the number of discs:
3

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

Total Number of moves are: 7
```

EXPERIMENT – 06

Design, Develop and Implement a menu driven Program in C for the following operations on QUEUE of Characters (Array Implementation of Queue with maximum size MAX)

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

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

PROGRAM CODE:

```
#include<stdio.h>
#include<conio.h>
int i,front,rear,ch,s[3],item;
void insert(),delet(),dis();
main()
{
front=0;
rear=-1;
ch=0;
while(ch!=4)
{
  printf("\nEnter the menu for queue operation\n");
  printf("\n1:insert\n2:delete\n3:display\n4:exit\n");
  printf("Input ur choice\n");
  scanf("%d",&ch);
  switch(ch)
          {
          case 1:
                insert(); break;
          case 2:
                delet(); break;
          case 3:
                dis(); break;
          case 4:
                exit(0); break;
         default:
                printf("Invalid choice\n");
           }
 }
```

```
return 0;
}
        void insert()
        {
          int item;
          if(rear==3-1)
          {
          printf("Queue is overflow\n");
          return;
          }
          printf("Enter an item to be inserted:");
          scanf("%d",&item);
          rear+=1;
          s[rear]=item;
        }
        void delet()
        {
        if(front>rear)
        {
          printf("Queue is empty\n");
          return;
        }
        printf("Item to be deleted is %d\n",s[front++]);
        }
        void dis()
        {
        if(front>rear)
        {
          printf("Queue is empty\n");
          return;
        }
        printf("\nQueue contains.....\n");
        for(i=front; i<=rear; i++)
        printf("%d\t",s[i]);
        }
```

OUTPUT:

```
Enter the menu for queue operation
1.Insert
2.Delete
3.Display
4.Exit
Input your choice  1
Enter an item to be inserted: 20

Enter the menu for queue operation
1.Insert
2.Delete
3.Display
4.Exit
Input your choice  1
Enter an item to be inserted: 40

Enter the menu for queue operation
1.Insert
2.Delete
3.Display
4.Exit
Input your choice 1
Enter an item to be inserted: 60

Enter the menu for queue operation
1.Insert
2.Delete
3.Display
4.Exit
Input your choice  3
Queue contains.....20 40 60

Enter the menu for queue operation
1.Insert
2.Delete
3.Display
4.Exit
Input your choice 2
Item to be deleted is  20

Enter the menu for queue operation
1.Insert
2.Delete
3.Display
4.Exit
Input your choice 2
Item to be deleted is  60

Enter the menu for queue operation
1.Insert
2.Delete
3.Display
4.Exit
Input your choice 3
Queue contains.....40
```

EXPERIMENT – 07

Design, Develop and Implement a menu driven Program in C for the following operations on Singly Linked List (SLL) of integer values

a. Create a SLL of N integers 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

```
#include <stdio.h>
#include<stdlib.h>
#include <malloc.h>
void create();
void insert();
void delet();
void display();
struct  node
{
int data;
struct node *link;
};
struct node *first=NULL,*last=NULL,*next,*prev,*cur;
void create()
{
  cur=(struct node*)malloc(sizeof(struct node));
  printf("\nENTER THE DATA: ");
  scanf("%d",&cur->data);
  cur->link=NULL;
  first=cur;
  last=cur;
}
void insert()
{
  int pos,c=1;
  cur=(struct node*)malloc(sizeof(struct node));
  printf("\nENTER THE DATA: ");
  scanf("%d",&cur->data);
  printf("\nENTER THE POSITION: ");
  scanf("%d",&pos);
  if((pos==1) &&(first!=NULL))
```

```
  {
   cur->link = first;
   first=cur;
  }
  else
  {
    next=first;
    while(c<pos)
    {
prev=next;
    next=prev->link;
    c++;
    }
    if(prev==NULL)
    {
    printf("\nINVALID POSITION\n");
    }
    else
    {
    cur->link=prev->link;
    prev->link=cur;
    }
 }
}
void delet()
{
int pos,c=1;
printf("\nENTER THE POSITION : ");
scanf("%d",&pos);
if(first==NULL)
{
printf("\nLIST IS EMPTY\n");
}
else if(pos==1 && first->link==NULL)
{
printf("\n DELETED ELEMENT IS %d\n",first->data);
free(first);
first=NULL;
}
else if(pos==1 && first->link!=NULL)
{
cur=first;
first=first->link;
cur->link=NULL;
printf("\n DELETED ELEMENT IS %d\n",cur->data);
```

```
free(cur);
}
else
{
next=first;
while(c<pos)
{
cur=next;
next=next->link;
c++;
}
cur->link=next->link;
next->link=NULL;
if(next==NULL)
{
printf("\nINVALID POSITION\n");
}
else
{
printf("\n DELETED ELEMENT IS %d\n",next->data);
free(next);
}
}
}
void display()
{
  cur=first;
  while(cur!=NULL)
  {
    printf("\n %d",cur->data);
    cur=cur->link;
  }
}
void main()
{
int ch;
printf("\n\nSINGLY LINKED LIST");
do
{
printf("\n\n1.CREATE\n2.INSERT\n3.DELETE\n4.EXIT");
printf("\n\nENTER YOUR CHOICE : ");
scanf("%d",&ch);
switch(ch)
{
case 1:
```

```
        create();
        display();
        break;
case 2:
        insert();
        display();
        break;
case 3:
        delet();
        display();
        break;
case 4:
        exit(0);
default:
   printf("Invalid choice...");
}
}while(1);
}
```