DATA STRUCTURES LAB PROGRAMS

STACK:

1. Stack using Arrays

Write a C program for the following operations on a stack of integers (use arrays).

a) Push
b) Pop
c) Display

```
#include<stdio.h>
#include<stdlib.h>
#define SIZE 5
int top=-1;
void push(int a[],int item)
{
    top=top+1;
    a[top]=item;
}
int pop(int a[])
{
    int item;
    item=a[top];
    top=top-1;
    return item;
}
void display(int a[])
{
    int i;
    if(top==-1)
        printf("The stack is empty\n");
    else if(top!=-1)
    {
        printf("The stack elements are\n");
        for(i=top; i>=0;i--)
            printf("%d ",a[i]);
        printf("\n");
    }
}

int main()
{
    int s[10],choice,item;
    while(1)
    {
        printf("Enter the choice\n");
        printf("1 Push\n2 Pop\n3 Display\n4 Exit\n");
        scanf("%d",&choice);
        switch(choice)
        {
            case 1: if(top==SIZE-1)
                    {
                        printf("The stack is full\n");
                        break;
                    }
                    else
                    {
                        printf("Enter the element to be pushed\n");
                        scanf("%d",&item);
                        push(s,item);
                    }
                    break;
            case 2: if(top==-1)
                    {
                        printf("The stack is empty\n");
                        break;
                    }
                    item=pop(s);
                    printf("Popped element is %d\n",item);
                    break;
            case 3: display(s);
                    break;
            case 4: exit(0);
        }
    }
    return 0;
}
```

2. Stack-Structure

Write a C program for the following operations on a stack of integers (use structure).

a) Push   b) Pop   c) Display

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

```
#define MAX 10
struct stack
{
    int top;
    int items[MAX];
};
void push(int,struct stack *);
void pop(struct stack *);
void display(struct stack *);
int main()
{
    struct stack s;
    s.top=-1;
    int choice,item;
    for(;;)
    {
        printf("Enter your choice\n");
        printf("1 Push\n2 Pop\n3 Display\n4 Exit\n");
        scanf("%d",&choice);
        switch(choice)
        {
            case 1: printf("Enter the item\n");
                    scanf("%d",&item);
                    push(item,&s);
                    break;
            case 2: pop(&s);
                    break;
            case 3: display(&s);
                    break;
            case 4: exit(0);
        }
    }
    return 0;
}
void push(int item,struct stack *s)
{
    if(s->top==MAX-1)
        printf("The stack is full\n");
    else
    {
        (s->top)++;
        s->items[s->top]=item;
    }
}
void pop(struct stack *s)
{
    int item;
    if(s->top==-1)
        printf("The stack is empty\n");
    else
    {
        item=s->items[s->top];
        (s->top)--;
        printf("%d deleted\n",item);
    }
}
void display(struct stack *s)
{
    int t=s->top;
    if(s->top==-1)
        printf("The stack is empty\n");
    else
    {
        printf("Elements in the stack are\n");
        while(t>-1)
        {
            printf("%d ",s->items[t--]);
        }
        printf("\n");
    }
}
```

3. String Palindrome using Stack

Write a C program to check given a string is a palindrome or not using stack.

```
#include<stdio.h>
#include<stdlib.h>
#define max 30
char stack[max], c[max], d[max];
int top=-1, j=0, k=0;
char pop();
```

```
void push(char);
int main()
{
    char a[30], b[30], sym;
    int m=0,i;
    printf("Enter a string\n");
    gets(a);
    if(a[0]=='\0')
    printf("Enter valid string\n");
    else
    {
        for(i=0;a[i]!='\0';i++)
        m++;
        for(i=0;i<m;i++)
        {
            sym=a[i];
            push(sym);
        }
        for(i=0;(i<m)&&(top!=-1);i++)
        b[i]=pop();
        b[i]='\0';
        d[k]='\0';
        printf("Reverse of given string is\n");
        printf("%s\n",d);
        for(i=0;b[i]!='\0';i++)
        {
            if(c[i]!=b[i])
            {
                printf("String is not a palindrome\n");
                exit(0);
            }
        }
        printf("String is a palindrome\n");
    }
    return 0;
}
void push(char sym)
{
    if(top==max-1)
    {
        printf("Stack is full\n");
        return;
    }
    top++;
    stack[top]=sym;
    if(sym!=' ')
    c[j++]=sym;
}
char pop()
{
    if(top==-1)
    {
        printf("Stack is empty\n");
        exit(0);
    }
    if(stack[top]==32)
        d[k++]=stack[top--];
    d[k++]=stack[top];
    return (stack[top--]);
}
```

4. Infix to postfix

*Write a C program to convert and print a given valid parenthesized infix arithmetic expression to postfix expression. The expression consists of single character operands and +,-, *, / operators.*
*Constraints: only four operators used +, -, *, / .*

```
#include<stdio.h>
#include<stdlib.h>
#include<math.h>
#include<ctype.h>
#include<string.h>
#define max 100
#define TRUE 1
#define FALSE 0
struct stack
{
        int top;
        char items[max];
};
struct stack s;
char infix[max],postfix[max];
```

```
int pos=0;
void convert();
void push(char);
char pop();
int precedence(char);
int f=0;
int empty();
int stackfull();
int main()
{
    s.top=-1;
    printf("Enter the infix expression\n");
    gets(infix);
    convert();
    if(f==0)
    {
        printf("The postfix expression is\n");
        puts(postfix);
    }
    return 0;
}
void convert()
{
    if(infix[0]==’\0’) { f=1; printf(“Invalid input\n”); return; }
    int i;
    char symbol,temp;
    for(i=0; infix[i]!='\0';i++)
    {
        symbol=infix[i];
        switch(symbol)
        {
            case '(': push(symbol);
                      break;
            case ')': while((temp=pop())!='(')
                          postfix[pos++]=temp;
                      break;
            case '+':
            case '-':
            case '*':
```

```
            case '/':
            case '$':
              while(!empty()&&precedence(s.items[s.top])>=precedence(symbol)&&precedence(symbol)!=-1)
                     {
                        temp=pop();
                        postfix[pos++]=temp;
                      }
                      push(symbol);
                      break;
            default: if(!isalpha(symbol))
                     {
                         printf("Invalid input\n");
                         f=1;
                         return;
                     }
                     else
                     {
                         postfix[pos++]=symbol;
                         break;
                     }
        }
    }
    while(!empty())
    {
        temp=pop();
        postfix[pos++]=temp;
    }
}
void push(char ele)
{
    if(stackfull())
        printf("Stack is full\n");
    else
        s.items[++s.top]=ele;
}
char pop()
{
    if(empty())
    {
        printf("Stack is empty\n");
        exit(0);
    }
    else
        return(s.items[s.top--]);
}
int stackfull() {
    if(s.top==max-1)
        return TRUE;
    else
        return FALSE;
}
int empty() {
    if(s.top==-1)
        return TRUE;
    else
        return FALSE;
}
int precedence(char symbol)
{
    switch(symbol)
    {
        case '$':return 3;
        case '*':
        case '/':return 2;
        case '+':
        case '-':return 1;
        case '(':
        case ')':return(0);
        default: printf("Invalid input\n");
                 return -1;
    }
}
```

5. Suffix-Postfix

Write a C program to evaluate a valid suffix/postfix expression using the stack. Assume that suffix/postfix expression is read as a single line consisting of non –negative single digit operands and binary arithmetic operators. The arithmetic operators are +, -,/,,^, ($).*

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

```
#include<ctype.h>
#include<math.h>
#include<string.h>
#define size 10
struct stack
{
    int top;
    double item[size];
};
double op(char,double,double);
void push(char, struct stack *);
double pop(struct stack *);
int main()
{
    int i;
    double op1,op2,res;
    struct stack s;    s.top = -1;
    char postfix[20],sym;
    printf("Enter the postfix expression\n");
    gets(postfix);
    for(i=0;i<strlen(postfix);i++)
    {
        sym=postfix[i];
        if(isdigit(sym))
            push(sym,&s);
        else
        {
            op2=pop(&s);
            op1=pop(&s);
            res=op(sym,op1,op2);
            if(res==-9999)
            {
                printf("Invalid input");
                return 0;
            }
            s.item[++s.top]=res;
        }
    }
    res=s.item[s.top--];
    printf("Result after evaluation is %.2f",res);
    return 0;
}
void push(char sym,struct stack *s)
{
    s->top++;
    s->item[s->top]=sym-'0';
}
double pop(struct stack *s)
{
    double ele;
    ele=s->item[s->top];
    s->top--;
    return(ele);
}
double op(char sym, double op1,double op2)
{
    switch(sym)
    {
        case '+':return(op1+op2);
        case '-':return(op1-op2);
        case '*':return(op1*op2);
        case '/':return(op1/op2);
        case '&':
        case '^':return(pow(op1,op2));
        default:return -9999;
    }
    exit(0);
}
```

QUEUE:

6.Queue

Write a C program to simulate the working of a queue of integers using structure. Provide the following operations

i) Insert   ii) Delete   iii) Display

```
#include<stdio.h>
#include<stdlib.h>
#define MAX 5
struct queue {
    int rear,front;
    int q[MAX];
};
void INSERT(int,struct queue ∗);
void DELETE(struct queue ∗);
void DISPLAY(struct queue ∗);
int main()
{
    int choice, item;
    struct queue s;
    s.rear=-1;
    s.front=0;
    for(;;)
    {
        printf("Enter your choice\n1 Insertion\n2 Deletion\n3 Display\n4 Exit\n");
        scanf("%d",&choice);
        switch(choice)
        {
            case 1:if(s.rear==MAX-1)
                        printf("Queue is full\n");
                        else
                        {
                            printf("Enter the element to be inserted\n");
                            scanf("%d",&item);
                            INSERT(item,&s);
                        }
                   break;
            case 2:DELETE(&s);
                   break;
            case 3:DISPLAY(&s);
                   break;
            case 4:exit(0);
        }
    }
    return 0;
}
void INSERT(int item,struct queue ∗s)
{
        s->rear=s->rear+1;
        s->q[s->rear]=item;
}
```

```
void DELETE(struct queue ∗s)
{
    int item;
    if(s->front>s->rear)
        printf("Queue is empty\n");
    else
    {
        item=s->q[s->front];
        printf("%d deleted\n",item);
        (s->front)++;
    }
}
void DISPLAY(struct queue ∗s)
{
    int i;
    if(s->front>s->rear)
        printf("Queue is empty\n");
    else
    {
        printf("Elements in the queue are\n");
        for(i=s->front;i<=s->rear;i++)
            printf("%d ",s->q[i]);
        printf("\n");
    }
}
```

7.Circular Queue

Write a C program to simulate the working of a circular queue of integers using array or structure. Provide the following operations.

i) Insert   ii) Delete   iii) Display

```
#include<stdio.h>
#include<stdlib.h>
#define SIZE 3
int items[SIZE];
int front=-1,rear=-1;
int isFull()
{
    if((front==rear+1)||(front==0 && rear==SIZE-1))
        return 1;
    return 0;
}
```

```
int isEmpty()
{
    if(front==-1)
        return 1;
    return 0;
}
void insert()
{
    int element;
    if(isFull())
        printf("Queue is full\n");
    else
    {
        printf("Enter the element\n");
        scanf("%d",&element);
        if(front==-1)
            front=0;
        rear=(rear+1)%SIZE;
        items[rear]=element;
    }
}
void delete()
{
    int element;
    if(isEmpty())
        printf("Queue is empty\n");
    else
    {
        element=items[front];
        if(front==rear) {front=-1; rear=-1;}
        else
            front=(front+1)%SIZE;
        printf("Deleted element %d \n",element);
    }
}
void display()
{
    int i;
    if(isEmpty())
```

```
        printf("Queue is empty\n");
    else
    {
        printf("Elements in the queue\n");
        for(i=front; i!=rear; i=(i+1)%SIZE)
            printf("%d ",items[i]);
        printf("%d \n",items[i]);

    }
}
int main()
{
    int choice;
    for(;;)
    {
        printf("Enter your choice\n1 Insertion\n2 Deletion\n3 Display\n4 Exit\n");
        scanf("%d",&choice);
        switch(choice)
        {
            case 1:insert();
                   break;
            case 2:delete();
                   break;
            case 3:display();
                   break;
            case 4:exit(0);
        }
    }
}
```

8.Priority Queue

Write a c program to design a priority queue.
Provide the following operations: i) Insert ii) Delete iii) Display

```
#include<stdio.h>
#include<stdlib.h>
#define MAX 5
int q[MAX];
int front=0, rear=-1,i,j;
void insert()
{
    printf("Enter value\n");
    int ele;
        scanf("%d",&ele);
    if(rear==MAX-1)
        printf("Queue overflow\n");
    else
    {
        j=rear;
        while((j>=front)&&(ele>q[j]))
        {
            q[j+1]=q[j];
            j--;
        }
        q[j+1]=ele;
        rear++;
    }
}

void delete()
{
    int flag=0;
    if(front>rear)
        printf("There are no elements to delete\n");
    else
    {
        int ele;
        printf("Enter value to delete\n");
        scanf("%d",&ele);
        for(i=front;i<=rear;i++)
        {
            if(q[i]==ele)
            {
                flag=1;
                for(j=i;j>front;j--)
                    q[j]=q[j-1];
                front++;
                break;
            }
        }
        if(flag==0)
            printf("%d not found\n",ele);
    }
}
void display()
{
    if(front>rear)
        printf("Queue is empty\n");
    else
    {
        for(i=front;i<=rear;i++)
            printf("%d ",q[i]);
            printf("\n");
    }
}

int main()
{
    int ch;
    printf("1 Insert\n2 Delete\n3 Display\n4 Exit\n");
    for(;;)
    {
    printf("Enter your choice\n");
    scanf("%d",&ch);
    switch(ch)
    {
        case 1:insert();
               break;
        case 2:delete();
               break;
        case 3:display();
               break;
        case 4:exit(0);
    }
    }
    return 0;
}
```

SINGLY LINKED LIST:

9. Stack using Singly Linked List

Write C program using dynamic variables and pointers to construct a singly Linked list to perform the operations of a stack of integers. (Push, Pop, Display).

i)use Structure to create a node

ii) make use of user-defined data type typedef.

```
#include<stdio.h>
#include<stdlib.h>
struct node
{
    int info;
    struct node *link;
};
typedef struct node * NODE;
NODE getnode()
{
    NODE X;
    X=(NODE)malloc(sizeof(struct node));
    return(X);
}
void freenode(NODE X)
{
    free(X);
}
NODE insert_front(NODE first,int item)
{
    NODE temp;
    temp=getnode();
    temp->info=item;
    temp->link=first;
    return(temp);
}
NODE delete_front(NODE first)
{
    NODE temp;
    if(first==NULL)
    {
        printf("List is empty\n");
        return first;
    }
    temp=first;
    temp=temp->link;
    printf("Deleted data is %d\n",first->info);
    freenode(first);
    return(temp);
}
void display(NODE first)
{
    NODE temp;
    if(first==NULL)
    {
        printf("List is empty\n");
        return;
    }
    printf("Contents of the Linked list are\n");
    temp=first;
    while(temp!=NULL)
    {
        printf("%d ",temp->info);
        temp=temp->link;
    }
    printf("\n");
}
int main()
{
    int choice,item;
    NODE first=NULL;
    for(;;)
    {
        printf("Enter your choice\n1 Insert rear\n2 Delete rear\n3 Display\n4 Exit\n");
        scanf("%d",&choice);
        switch(choice)
        {
            case 1:printf("Enter the item\n");
                   scanf("%d",&item);
                   first=insert_front(first,item);
                   break;
            case 2:first=delete_front(first);
                   break;
            case 3:display(first);
```

```
                        break;
                case 4:exit(0);
            }
        }
}
```

10. Queue using Singly Linked List

Write a C program using dynamic variables and pointers to construct a singly Linked list to perform the operations of a queue of integers.

```
#include<stdio.h>
#include<stdlib.h>
struct node
{
    int info;
    struct node *link;
};
typedef struct node * NODE;
NODE getnode()
{
    NODE X;
    X=(NODE)malloc(sizeof(struct node));
    if(X==NULL)
    {
        printf("Memeory not available\n");
    }
    return(X);
}
void freenode(NODE X)
{
    free(X);
}
NODE insert_rear(int item,NODE first)
{
    NODE temp,cur;
    temp=getnode();
    temp->info=item;
    temp->link=NULL;
    if(first==NULL)
        return (temp);
    cur=first;
    while(cur->link!=NULL)
    {
        cur=cur->link;
    }
    cur->link=temp;
    return (first);
}
NODE delete_front(NODE first)
{
    NODE temp;
    if(first==NULL)
    {
        printf("List is empty\n");
        return(first);
    }
    temp=first;
    temp=temp->link;
    printf("Deleted data is %d\n",first->info);
    freenode(first);
    return(temp);
}

void display(NODE first)
{
    NODE temp;
    if(first==NULL)
    {
        printf("List is empty\n");
        return;
    }
    printf("Contents of the Linked list are\n");
    temp=first;
    while(temp!=NULL)
    {
        printf("%d ",temp->info);
        temp=temp->link;
    }
    printf("\n");
}
```

```
int main()
{
    int choice,item;
    NODE first=NULL;
    for(;;)
    {
        printf("Enter your choice");
        printf("\n1 Insert rear\n2 Delete front\n3 Display\n4 Exit\n");
        scanf("%d",&choice);
        switch(choice)
        {
            case 1:printf("Enter the data\n");
                   scanf("%d",&item);
                   first=insert_rear(item,first);
                   break;
            case 2:first=delete_front(first);
                   break;
            case 3:display(first);
                   break;
            case 4:exit(0);
        }
    }
    return 0;
}
```

11. Singly Linked List-Dynamic

Write a C program using dynamic variables and pointers to construct a singly linked list consisting of the following information in each node: student id (integer), student name (character string) and semester (integer). The operations to be supported are:
i) Insert at specified position
ii) Delete front
iii) Display

```
#include<stdio.h>
#include<stdlib.h>
int count=0;
struct node
{
    int student_id;
    int semester;
    char student_name[20];
    struct node *link;
};
void insert(struct node ∗∗s,int pos)
{
    int i;
     if(pos>count+1)
    {
        printf("Invalid position\n");
        return;
    }
    struct node ∗newnode;
    newnode=(struct node ∗)malloc(sizeof(struct node));
    printf("Enter id\n");
    scanf("%d",&(newnode->student_id));
    printf("Enter name\n");
    scanf("%s",newnode->student_name);
    printf("Enter semester\n");
    scanf("%d",&(newnode->semester));
    struct node ∗temp=∗s;
    if((∗s)==NULL)
    {
        (∗s)=newnode;
        count++;
    }
    else if(pos==0 || pos==1)
    {
        count++;
        newnode->link=∗s;
        ∗s=newnode;
    }
    else
    {
    for(i=2; i<pos; i++)
    {
        temp=temp->link;
    }
    newnode->link=temp->link;
    temp->link=newnode;
    count++;
   }
}
```

```
void delete_front(struct node **s)
{
    if((*s)==NULL)
    {
        printf("List is empty\n");
        return;
    }
    struct node *temp=*s;
    struct node *cur=temp->link;
    *s=cur;
    int item=temp->student_id;
    free(temp);
    printf("Deleted ID is %d\n",item);
}
void display(struct node *s)
{
    if(s==NULL)
    {
        printf("List is empty\n");
        return;
    }
    struct node *temp=s;
    printf("Contents are\n");
    while(temp!=NULL)
    {
        printf("ID %d\n",temp->student_id);
        printf("Name %s\n",temp->student_name);
        printf("Semester %d\n",temp->semester);
        temp=temp->link;
    }
}
int main()
{
    int choice,pos;
    struct node *s=NULL;
    for(;;)
    {
        printf("Enter your choice\n1 Insert at position\n2 Delete front\n3 Display\n4 Exit\n");
        scanf("%d",&choice);
        switch(choice)
        {
            case 1: printf("Enter the position\n");
                    scanf("%d",&pos);
                    insert(&s,pos);
                    break;
            case 2: delete_front(&s);
                    break;
            case 3:
                    display(s);
                    break;
            case 4: exit(0);
        }
    }
    return 0;
}
```

12.Circular Linked List

Write a C program to support the following operations on a circular linked list where each node consists of integers.

1. Insert rear
2. Delete rear
3. Display.

```
#include<stdio.h>
#include<stdlib.h>
struct node
{
    int data;
    struct node *link;
};
typedef struct node * NODE;
NODE insert_rear(NODE first,int item)
{
NODE newnode;
newnode=(NODE)malloc(sizeof(struct node));
newnode->data=item;
if(first==NULL)
{
     first=newnode;
     first->link=newnode;
```

```
      return first;
}
else
{
      NODE temp=first;
      while(temp->link!=first)
      temp=temp->link;
      temp->link=newnode;
      newnode->link=first;
      return first;
}
}
NODE delete_rear(NODE first)
{
     if(first==NULL)
     {
            printf("List is empty\n");
            return first;
     }
     else
     {
          int itm;
          NODE temp=first;
          if(first->link==first)
          {
               itm=first->data;
               free(temp);
               first=NULL;
          }
         else
          {
                NODE prev=NULL;
                while(temp->link!=first)
                {
                     prev=temp;
                     temp=temp->link;
                }
                prev->link=first;
                itm=temp->data;
```

```
                    free(temp);
                }
                printf("Deleted data is %d\n",itm);
                return first;
}
}
void display(NODE first)
{
        if(first==NULL)
        {
                printf("List is empty");
                return;
        }
        else
        {
                NODE temp;
                temp=first;
                printf("Contents of the Circular Linked list are\n");
                while(temp->link!=first)
                {
                        printf("%d ",temp->data);
                        temp=temp->link;
                }
                printf("%d",temp->data);
         }
}
int main()
{
        int choice,val;
        NODE first=NULL;
        for(;;)
        {
                printf("Enter your choice\n1 Insert rear\n2 Delete rear\n3 Display\n4 Exit\n");
                scanf("%d",&choice);
                switch(choice)
                {
                        case 1: printf("Enter the data\n");
                                scanf("%d",&val);
                                first=insert_rear(first,val);
```

```
                     break;
             case 2: first=delete_rear(first);
                     break;
             case 3:display(first);
                   printf("\n");
                    break;
            case 4:exit(0);
    }
}
    return 0;
}
```

DOUBLY LINKED LIST:

13.Doubly linked list

Write a C program to support the following operations on a doubly linked list where each node consists of integers.

i) Insert front ii) Delete rear iii) Display.

```
#include<stdio.h>
#include<stdlib.h>
struct node
{
    int info;
    struct node *llink, *rlink;
};
typedef struct node * NODE;
NODE insert_front(NODE first,int val)
{
    NODE newnode;
    newnode=(NODE)malloc(sizeof(struct node));
    newnode->info=val;
    newnode->llink=NULL;
    if(first==NULL)
    {
        first=newnode;
        newnode->rlink=NULL;
        return first;
    }
    else
    {
        newnode->rlink=first;
        first->llink=newnode;
        return newnode;
    }
}
NODE delete_rear(NODE first)
{
    if(first==NULL)
    {
        printf("List is Empty\n");
        return first;
    }
    int del;
    NODE cur=first;
    if(first->rlink==NULL)
    {
        del=first->info;
        free(first);
        first=NULL;
    }
    else
    {
    NODE prev=NULL;
    while(cur->rlink!=NULL)
    {
        cur=cur->rlink;
    }
    prev=cur->llink;
    prev->rlink=NULL;
    del=cur->info;
    free(cur);
    }
    printf("Deleted data is %d\n",del);
    return first;
}
void display(NODE first)
{
    if(first==NULL)
    {
        printf("List is Empty\n");
        return;
    }
    else
    {
        NODE temp=first;
        while(temp!=NULL)
        {
            printf("%d ",temp->info);
            temp=temp->rlink;
        }
        printf("\n");
    }
}

int main()
{
    NODE first=NULL;
    int choice,val;
    while(1)
    {
        printf("Enter your choice\n1 Insert front\n2 Delete rear\n3 Display\n4 Exit\n");
        scanf("%d",&choice);
        switch(choice)
        {
            case 1: printf("Enter the data\n");
                scanf("%d",&val);
                first=insert_front(first,val);
                break;
            case 2: first=delete_rear(first);
                break;
            case 3: display(first);
                break;
            case 4: exit(0);
        }
    }
    return 0;
}
```

14. Circular Doubly Linked List I

Write a C program to support the following operations on a circular doubly linked list (with or without header node where each node consists of integers.

i) Insert front ii) Delete rear iii) Display.

```
#include<stdio.h>
#include<stdlib.h>
struct node
{
    int info;
    struct node *llink, *rlink;
};
typedef struct node * NODE;
NODE insert_front(NODE first,int val)
{
    NODE newnode;
    newnode=(NODE)malloc(sizeof(struct node));
    newnode->info=val;
    if(first==NULL)
    {
        first=newnode;
        newnode->llink=newnode;
        newnode->rlink=newnode;
        return first;
    }
    else
    {
        NODE last=first->llink;
        newnode->rlink=first;
        newnode->llink=last;
        last->rlink=newnode;
        first->llink=newnode;
        return newnode;
    }
}
NODE delete_rear(NODE first)
{
    if(first==NULL)
    {
        printf("List is Empty\n");
        return first;
    }
    int del;
    NODE cur=first;
    if(first->rlink==first)
    {
        del=first->info;
        free(first);
        first=NULL;
    }
    else
    {
    NODE prev=NULL;
    cur=first->llink;
    prev=cur->llink;
    prev->rlink=first;
    first->llink=prev;
    del=cur->info;
    free(cur);
    }
    printf("Deleted data is %d\n",del);
    return first;
}
void display(NODE first)
{
    if(first==NULL)
    {
        printf("List is Empty\n");
        return;
    } else
    {
        NODE temp=first;
        while(temp->rlink!=first)
        {
            printf("%d ",temp->info);
            temp=temp->rlink;
        }
        printf("%d \n",temp->info);
    }
}
int main()
{
    NODE first=NULL;
    int choice,val;
    while(1)
    {
        printf("Enter your choice\n1 Insert front\n2 Delete rear\n3 Display\n4 Exit\n");
        scanf("%d",&choice);
        switch(choice)
        {
            case 1: printf("Enter the data\n");
                scanf("%d",&val);
                first=insert_front(first,val);
                break;
            case 2: first=delete_rear(first);
                break;
            case 3: display(first);
                break;
            case 4: exit(0);
        }
    }
    return 0;
}
```

15.Circular doubly linked list II

Write a C program to support the following operations on a circular doubly linked list (with or without header node) where each node consists of integers.

i) Insert rear
ii) Delete front
iii) Insert Right
iv) Display

```
#include<stdio.h>
#include<stdlib.h>
struct node
{
    int info;
    struct node *llink, *rlink;
};
typedef struct node * NODE;
NODE insert_rear(NODE head,int val)
{
    NODE newnode;
    newnode=(NODE)malloc(sizeof(struct node));
    newnode->info=val;
    if(head->rlink==head)
    {
        head->rlink=newnode;
        head->llink=newnode;
        newnode->llink=head;
        newnode->rlink=head;
        return head;
    }
    else
    {
        NODE temp=head->llink;
        temp->rlink=newnode;
        newnode->llink=temp;
        newnode->rlink=head;
        head->llink=newnode;
        return head;
    }
}
NODE delete_front(NODE head)
{
    int del;
    NODE temp=head->rlink;
    if(head->rlink==head)
    {
        printf("List is empty\n");
        return head;
    }
    else
    {
        NODE next=temp->rlink;
        head->rlink=next;
        next->llink=head;
        del=temp->info;
        free(temp);
       printf("%d deleted\n",del);
       return head;
    }
}
void display(NODE head)
{
    if(head->rlink==head)
```

```
        {
            printf("List is empty\n");
            return;
        }
        else
        {
            printf("Contents of the Circular Doubly Linked list are\n");
            NODE temp=head->rlink;
            while(temp!=head)
            {
                printf("%d ",temp->info);
                temp=temp->rlink;
            }
            printf("\n");
        }
}
NODE insert_right(NODE head,int key,int val)
{
        NODE temp=head->rlink;
        while(temp!=head)
        {
            if(temp->info==key)
                break;
            temp=temp->rlink;
        }
        if(temp==head)
        {
            printf("Key not found\n");
            return head;
        }
        else
        {
        NODE newnode;
        newnode=(NODE)malloc(sizeof(struct node));
        newnode->info=val;
        NODE next=temp->rlink;
        temp->rlink=newnode;
        newnode->llink=temp;
        newnode->rlink=next;
        next->llink=newnode;
```

```
    return head;
    }
}
int main()
{
    NODE head=(NODE)malloc(sizeof(struct node));
    head->info=0;
    head->rlink=head;
    head->llink=head;
    int choice,val,key;
    for(;;)
    {
        printf("Enter your choice\n1 Insert rear\n2 Delete front\n3 Insert right\n4 Display\n5 Exit\n");
        scanf("%d",&choice);
        switch(choice)
        {
            case 1:printf("Enter the data\n");
                   scanf("%d",&val);
                   head=insert_rear(head,val);
                   break;
            case 2: head=delete_front(head);
                   break;
            case 3: printf("Enter the key\n");
                  scanf("%d",&key);
                  printf("Enter a node\n");
                  scanf("%d",&val);
                  head=insert_right(head,key,val);
                  break;
            case 4:display(head);
                  break;
            case 5:exit(0);
                  break;
            default:printf("Invalid input\n");
       }
    }
    return 0;
}
```

TREES:

16.Binary Tree

Write a C program to implement a binary tree of integers and perform the following traversal techniques.
i)In-order traversal
ii)Post-order traversal
iii)Pre-order traversal.

```
#include<stdio.h>
#include<stdlib.h>
struct node
{
    int info;
    struct node *lchild;
    struct node *rchild;
};
typedef struct node * NODE;
NODE create_B_Tree()
{
    NODE newnode;
    int data=0;
    newnode=NULL;
    printf("Enter data ('0' if no data)");
    scanf("%d",&data);
    if(data)
    {
        newnode=(NODE)malloc(sizeof(struct node));
        newnode->info=data;
        printf("\nLeft child of %d\n",newnode->info);
        newnode->lchild=create_B_Tree();
        printf("\nRight child of %d\n",newnode->info);
        newnode->rchild=create_B_Tree();
    }
    return newnode;
}
void pre_order(NODE root)
{
    if(root!=NULL)
    {
        printf("%d\n",root->info);
        pre_order(root->lchild);
        pre_order(root->rchild);
    }
}
void post_order(NODE root)
{
    if(root!=NULL)
    {
        post_order(root->lchild);
        post_order(root->rchild);
        printf("%d\n",root->info);
    }
}
void in_order(NODE root)
{
    if(root!=NULL)
    {
        in_order(root->lchild);
        printf("%d\n",root->info);
        in_order(root->rchild);
    }
}
int main()
{
    printf("Create binary tree, start from root\n");
    NODE root;
    root=create_B_Tree();
    while(1)
    {
        printf("Select mode of traversal for displaying the binary tree\n");
        printf("1 Pre-order\n2 Post-order\n3 In-order\n4 Exit\nChoice \n");
        int choice;
        scanf("%d",&choice);
        switch(choice)
        {
            case 1:pre_order(root);
                printf("\n");
                break;
            case 2:post_order(root);
                printf("\n");
                break;
            case 3:in_order(root);
            printf("\n");
                break;
            case 4:exit(0);
        }
    }
    return 0;
}
```

17.Expression Tree

Write a C program to construct expression tree for the given postfix expression and evaluate the same.

```
#include<stdio.h>
#include<stdlib.h>
#include<ctype.h>
#define max 100
struct node
{
    int data;
    struct node *llink,*rlink;
};
typedef struct node * NODE;
NODE construct_tree(char postfix[max])
{
    if(postfix[0]==’\0’) return NULL;
    char sym;
    NODE newnode,stack[max];
    int i,top=-1;
    for(i=0;postfix[i]!='\0';i++)
    {
        sym=postfix[i];
        newnode=(NODE)malloc(sizeof(struct node));
        newnode->data=sym;
        newnode->llink=NULL;
        newnode->rlink=NULL;
        if(isdigit(sym))
        {
            stack[++top]=newnode;
            continue;
        }
        switch(sym)
        {
            case '+':
            case '-':
            case '*':
            case '/': newnode->rlink=stack[top--];
                      newnode->llink=stack[top--];
                      stack[++top]=newnode;
                      break;
            default: return NULL;
        }
    }
    return stack[top--];
}
float evaluate(NODE root)
{
    switch(root->data)
    {
        case '+':return(evaluate(root->llink)+evaluate(root->rlink));
        case '-':return(evaluate(root->llink)-evaluate(root->rlink));
        case '*':return(evaluate(root->llink)*evaluate(root->rlink));
        case '/':return(evaluate(root->llink)/evaluate(root->rlink));
        default: return(root->data - '0');
    }
}
int main()
{
    float res;
    char postfix[max];
    NODE root=NULL;
    printf("Enter the postfix expression\n");
    gets(postfix);
    root=construct_tree(postfix);
    if(root==NULL)
    {
        printf("Invalid input\n");
        return 0;
    }
    res=evaluate(root);
    printf("Result after evaluation is %.2f\n",res);
    return 0;
}
```

18.Binary-Search Tree

Write a C program to implement a binary search tree of integers and perform the following traversal techniques:
i) In-order traversal
ii)Find the maximum element
iii) Search an element
Display "Duplication is not allowed" if same element is inserted again.
Display "Key element not found" if searched element is not present in the tree.
Display "Search is successful" if searched element is present in the tree.
Displpay "Maximum element is 15" if the maximum element is 15 in the tree.

```
#include<stdio.h>
#include<stdlib.h>
struct node
{
    int data;
    struct node *llink,*rlink;
};
typedef struct node * NODE;
NODE create_node(int val)
{
    NODE newnode;
    newnode=(NODE)malloc(sizeof(struct node));
    newnode->data=val;
    newnode->llink=NULL;
    newnode->rlink=NULL;
    return newnode;
}
void setleft(NODE p,int val)
{
    if(p==NULL)
        printf("Insertion is not possible\n");
    else if(p->llink!=NULL)
        printf("Invalid insertion\n");
    else
        p->llink=create_node(val);
}
void setright(NODE p,int val)
{
    if(p==NULL)
        printf("Insertion is not possible\n");
    else if(p->rlink!=NULL)
```

```
        printf("Invalid insertion\n");
    else
        p->rlink=create_node(val);
}
NODE create_BS_Tree(NODE root,int val)
{
    if(root==NULL)
    {
        root=create_node(val);
    }
    else
    {
    NODE p,q;
    p=q=root;
    while(val!=p->data&&q!=NULL)
    {
        p=q;
        if(val<p->data)
            q=p->llink;
        else
            q=p->rlink;
    }
    if(val==p->data)
    {
        printf("Duplication is not allowed\n");
    }
    else if(val<p->data)
        setleft(p,val);
    else
        setright(p,val);
    }
    return root;
}
void inorder(NODE root)
{
    if(root!=NULL)
    {
        inorder(root->llink);
        printf("%d ",root->data);
        inorder(root->rlink);
    }
}
void search(NODE root,int key)
{
    NODE p,q;
    p=q=root;
    while(key!=p->data&&q!=NULL)
    {
        p=q;
        if(key<p->data)
            q=p->llink;
        else
            q=p->rlink;
    }
    if(key==p->data)
        printf("Search is successful\n");
    else
        printf("Key element not found\n");
}
int maximum(NODE root)
{
    while(root->rlink!=NULL)
    {
        root=root->rlink;
    }
    return root->data;
}
int main()
{
    int choice, val,key,max;
    NODE root=NULL;
    printf("1 Insert\n2 In-order\n3 Search\n4 Maximum\n5 Exit\n");
    while(1)
    {
        printf("Enter your Choice\n");
        scanf("%d",&choice);
        switch(choice)
        {
            case 1:printf("Enter the element\n");
                   scanf("%d",&val);
                   root=create_BS_Tree(root,val);
                   break;
            case 2:inorder(root);
                   printf("\n");
                   break;
            case 3:printf("Enter the element to be searched\n");
                   scanf("%d",&key);
                   search(root,key);
                   break;
            case 4:max=maximum(root);
                   printf("Maximum element is %d\n",max);
                   break;
            case 5:exit(0);
                   break;
            default:printf("Invalid choice\n");
        }
    }
    return 0;
```

***