DAY-3 (Data-Structures) Lab Session

EXP - 1

TITLE : C Program to Implement Linked List Data Structure

PROGRAM :

```
#include <stdio.h>
#include <stdlib.h>
struct lis
{
        int data;
        struct lis *next;
};
struct lis *head;
int main()
{
        int a,b,c;
        while(true)
        {

        printf("1.HEAD\n2.INSERTLAST\n3.INSERTFIRST\n4.INSERTMIDDLE\n5.DELETEFIRST\n6
.DELETEMIDDLE\n7.DELETELAST\n8.PRINT\n");
                printf("Enter The Option:");
                scanf("%d",&a);
        switch(a)
        {
                case 1:
                        struct lis *nn;
                        nn=(struct lis*)malloc(sizeof(struct lis));
                        if(head==NULL)
                        {
                                printf("Enter The data:");
                                scanf("%d",&b);
                                nn->data=b;
                                nn->next=NULL;
                                head=nn;
                        }
                        break;
            case 2:
```

```
else
                {
                        struct lis *nn;
                        struct lis *h;
                        nn=(struct lis*)malloc(sizeof(struct lis));
                        h=(struct lis*)malloc(sizeof(struct lis));
                        h=head;
                        printf("Enter the data:");
                        scanf("%d",&b);
                        nn->data=b;
                        nn->next=NULL;
                        while(h->next!=NULL)
                        {
                                h=h->next;
                        }
                        h->next=nn;
            }
                break;
        case 3:
                if(head==NULL)
                {
                        printf("List Empty\n");
                }
                else
                {
                        struct lis *nn;
                        nn=(struct lis*)malloc(sizeof(struct lis));
                        printf("Enter the data:");
                        scanf("%d",&b);
                        nn->data=b;
                        nn->next=head;
                        head=nn;
                }
                break;
        case 4:
                if(head==NULL)
```

```
        else
        {
                struct lis *nn,*temp,*prev;
                temp=head;
                nn=(struct lis*)malloc(sizeof(struct lis));
                int key;
                printf("Enter The Key to Insert After:");
                scanf("%d",&c);
                printf("Enter The New Node Element to Insert:");
                scanf("%d",&b);
                nn->data=b;
                while(temp->data!=c)
                {
                        temp=temp->next;
                        prev=temp->next;
                }
                temp->next=nn;
                nn->next=prev;
        }
        break;
case 5:
        struct lis *v;
        v=(struct lis*)malloc(sizeof(struct lis));
        v=head->next;
        head->next=NULL;
        head=v;
        break;
case 6:
        struct lis *prev,*temp;
        printf("Enter The Data To be Deleted:");
        scanf("%d",&c);
        temp=head;
        while(temp->data!=c)
        {
                prev=temp;
```

```
                temp=temp->next;
            }
            temp->next=prev->next;
            break;
        case 7:
            struct lis *tem,*pre;
            tem=head;
            while(tem->next!=NULL)
            {
                pre=tem;
                tem=tem->next;
            }
            pre->next=NULL;
            break;
        case 8:
            struct lis *h;
            h=(struct lis*)malloc(sizeof(struct lis));
            h=head;
            while(h->next!=NULL)
            {
                printf("[%d|.]-> ",h->data);
                h=h->next;
            }
            printf("[%d|/]\n",h->data);
            break;
    }
}
}
```

INPUT AND OUTPUT :

```
Enter The data:8
1.HEAD
2.INSERTLAST
3.INSERTFIRST
4.INSERTMIDDLE
5.DELETEFIRST
6.DELETEMIDDLE
7.DELETELAST
8.PRINT
Enter The Option:3
Enter the data:9
1.HEAD
2.INSERTLAST
3.INSERTFIRST
4.INSERTMIDDLE
5.DELETEFIRST
6.DELETEMIDDLE
7.DELETELAST
8.PRINT
Enter The Option:8
[9|.]-> [8|/]
```

RESULT :

The C Program for Implementing Linked List Data Structure is Compiled and Executed Using Dev-C++ and the Output is Verified.

DAY-3 (Data-Structures) Lab Session

EXP - 2

TITLE : C Program to Merge two Linked List Data Structures

PROGRAM :

```
#include <stdio.h>
#include <stdlib.h>
struct lis
{
        int data;
        struct lis *next;
};
struct lis *h1,*h2;
int main()
{
        int a,b,c,i,j,k=0;
        while(k==0)
        {
                printf("1.INSERT1\n2.INSERT2\n");
                printf("Enter Choice:");
                scanf("%d",&b);
                switch(b)
                {
                        case 1:
                                printf("Enter -1 to stop Storing\n");
                                printf("Enter The Data:");
                                scanf("%d",&c);
                                if(c!=-1)
                                {
                                        struct lis *nn;
                                        nn=(struct lis*)malloc(sizeof(struct lis));
                                        nn->data=c;
                                        nn->next=NULL;
                                        h1=nn;
                                        while(c!=-1)
                                        {
                                                struct lis *h;
```

```
h=h1;
                    }
                    break;
            case 2:
                    printf("Enter -1 to stop Storing\n");
                    printf("Enter The Data:");
                    scanf("%d",&c);
                    if(c!=-1)
                    {
                            struct lis *nn;
                            nn=(struct lis*)malloc(sizeof(struct lis));
                            nn->data=c;
                            nn->next=NULL;
                            h2=nn;
                            while(c!=-1)
                            {
                                    struct lis *h;
                                    struct lis*n;
                                    n=(struct lis*)malloc(sizeof(struct lis));
                                    h=h2;
                                    printf("Enter The Data:");
                                    scanf("%d",&c);
                                    n->data=c;
                                    n->next=NULL;
                                    while(h->next!=NULL)
                                    {
                                            h=h->next;
                                    }
                                    h->next=n;
                            }
                            struct lis *h;
                            h=h2;
                            k=1;
                    }
                    break;
```

```
            }
        }
        struct lis *h,*l,*t;
        h=h1;
        while(h->next!=NULL)
        {
            t=h;
            h=h->next;
        }
        t->next=h2;
        h=h1;
        while(h->next!=NULL)
        {
            printf("[%d|.]->",h->data);
            h=h->next;
        }
        printf("Merged LinkedList");
}
```

INPUT AND OUTPUT :

```
1.INSERT1
2.INSERT2
Enter Choice:1
Enter -1 to stop Storing
Enter The Data:5
Enter The Data:9
Enter The Data:4
Enter The Data:3
Enter The Data:
8
Enter The Data:-1
1.INSERT1
2.INSERT2
Enter Choice:2
Enter -1 to stop Storing
Enter The Data:3
Enter The Data:4
Enter The Data:7
Enter The Data:-1
[5|.]->[9|.]->[4|.]->[3|.]->[8|.]->[3|.]->[4|.]->[7|.]->Merged LinkedList
```

RESULT :

The C Program for Merging Linked List Data Structures is Compiled and Executed Using Dev-C++ and the Output is Verified.

DAY-3 (Data-Structures) Lab Session

EXP - 3

TITLE : C Program for Performing Stack Operation.

PROGRAM :

```
#include <stdio.h>
int main()
{
        int b,top=-1,c,e,i;
        printf("Enter The Size of the Stack:");
        scanf("%d",&b);
        int a[b];
        while(true)
        {
                printf("1.PUSH\n2.POP\n3.SEEK\n4.PRINT\n");
                printf("Enter The Operation Number:");
                scanf("%d",&c);
                switch(c)
                {
                        case 1 :
                                if(top==b-1)
                                {
                                        printf("\nStack Overflow!!\n");
                                }
                                else
                                {
                                printf("Enter The Element to Push:");
                                scanf("%d",&e);
                                top+=1;
                                a[top]=e;
                            }
                            break;
                        case 2 :
                                if(top==-1)
                                {
                                        printf("\nStack Underflow!!!\n\n");
                                }
```

```
        else
        {
                top=top-1;
                printf("\npopped\n\n");
        }
        break;
case 3 :
        if(top==-1)
        {
                printf("\nEmpty Stack!!!\n\n");
        }
        else
        {
                printf("\n%d=top\n\n",a[top]);
        }
        break;
case 4 :
        if(top==-1)
        {
                printf("\nEmpty Stack\n\n");
        }
        else
        {
                for(i=top;i>=0;i--)
                {
                        if(i==top)
                        {
                                printf("\n\n%d-->top\n",a[i]);
                        }
                        else
                        {
                                printf("%d\n",a[i]);
                        }
                }
                printf("\n\n\n");
        }
```

```
                    break;
            default :
                    printf("Enter the Valid Operation to Proceed");
        }
    }
}
```

INPUT AND OUTPUT :

```
Enter The Size of the Stack:5
1.PUSH
2.POP
3.SEEK
4.PRINT
Enter The Operation Number:1
Enter The Element to Push:9
1.PUSH
2.POP
3.SEEK
4.PRINT
Enter The Operation Number:1
Enter The Element to Push:5
1.PUSH
2.POP
3.SEEK
4.PRINT
Enter The Operation Number:4


5-->top
9
```

RESULT :

The C Program for Implementing Stack Operations is Compiled and Executed Using Dev-C++ and the Output is Verified.

DAY-3 (Data-Structures) Lab Session

EXP - 4

TITLE : C Program for Performing Queue Operation.

PROGRAM :

```
#include <stdio.h>
int main()
{
        int b,f=-1,r=-1,c,e,i;
        printf("Enter the Size of the Queue:");
        scanf("%d",&b);
        int a[b];
        while(true)
        {
                printf("1.ENQUEUE\n2.DEQUEUE\n3.PRINT\n4.CLEAR\n");
                printf("Enter the Operation Number to Proceed:");
                scanf("%d",&c);
                switch(c)
                {
                        case 1:
                                if(r==b-1)
                                {
                                        printf("\nQueue Full\n\n");
                                }
                                else
                                {
                                        printf("Enter The Element to Enqueue:");
                                        scanf("%d",&e);
                                        if(f==-1)
                                        {
                                                f+=1;
                                                r+=1;
                                                a[r]=e;
                                        }
                                        else
                                        {
                                                r=r+1;
```

```
                }
            }
            break;
        case 2:
            if(f==-1 || f==b)
            {
                printf("\nQUEUE EMPTY\n\n");
            }
            else
            {
                f+=1;
                printf("\nDequeued!!\n\n");
            }
            break;
        case 3:
            if(f==-1 || f==b)
            {
                printf("\nQUEUE EMPTY\n\n");
            }
            else
            {
                for(i=r;i>=f;i--)
                {
                    if(i==r)
                    {
                        if(i==f)
                        {
                            printf("\n%d-->rear&front\n",a[i]);
                        }
                        else
                        {
                            printf("\n%d-->rear\n",a[i]);
                        }
                    }
                    else if(i==f)
                    {
```

```
                        {
                                printf("%d--front\n\n",a[i]);
                        }
                        else
                        {
                                printf("%d\n",a[i]);
                        }
                }
        }
        break;
case 4:
        f=-1;
        r=-1;
        break;
default :
        printf("\nEnter Valid Operation\n\n");;)}}
```

INPUT AND OUTPUT :

```
Enter the Size of the Queue:5
1.ENQUEUE
2.DEQUEUE
3.PRINT
4.CLEAR
Enter the Operation Number to Proceed:1
Enter The Element to Enqueue:5
1.ENQUEUE
2.DEQUEUE
3.PRINT
4.CLEAR
Enter the Operation Number to Proceed:3

5-->rear&front
```

RESULT :

The C Program for Implementing Queue Operations is Compiled and Executed Using Dev-C++ and the Output is Verified.

DAY-3 (Data-Structures) Lab Session

EXP - 5

TITLE : C Program for Converting Infix Notation to Postfix Notation

PROGRAM :

```
#include <stdio.h>
#include <string.h>
main()
{
        char a[100],p[100];
        int t=-1,i,j,l;
        printf("Enter The Expression:");
        scanf("%s",a);
        l=strlen(a);
        a[l]='#';
        for(i=0;i<=l;i++)
        {
          if(a[i]>='a' && a[i]<='z')
                {
                        printf("%c",a[i]);
                }
                else if(a[i]=='*' || a[i]=='/')
                {
                        if(t==(-1))
                        {
                                t+=1;
                                p[t]=a[i];
                        }
                        else
                        {
                                if(p[t]=='*'||p[t]=='/')
                                {
                                        while(p[t]=='*'||p[t]=='/')
                                        {
                                                printf("%c",p[t]);
                                                t-=1;
                                        }
```

```
p[t]=a[i];
                        }
                }
        }
        else if(a[i]=='+' || a[i]=='-')
        {
                if(t==-1)
                {
                        t+=1;
                        p[t]=a[i];
                }
                else
                {
                        if(p[t]=='+'||p[t]=='-'||p[t]=='*'||p[t]=='/')
                        {
                                while(p[t]=='+'||p[t]=='-'||p[t]=='*'||p[t]=='/')
                                {
                                        printf("%c",p[t]);
                                        t-=1;
                                }
                                t+=1;
                                p[t]=a[i];
                        }
                        else
                        {
                                t+=1;
                                p[t]=a[i];
                        }
                }
        }
        else if(a[i]=='(')
        {
                t+=1;
                p[t]=a[i];
        }
        }
```

```
else if(a[i]==')')
{
        while(p[t]!='(')
        {
                printf("%c",p[t]);
                t-=1;
        }
        t-=1;
}
else if(a[i]=='#')
{
        if(t>-1)
        {
        for(j=t;j>=0;j--)
        {
                printf("%c",p[j]);}}}}}
```

INPUT AND OUTPUT :

```
Enter The Expression:a+b*c
abc*+
```

RESULT :

The C Program for Converting Infix to Postfix Notation using Stack is Compiled and Executed Using Dev-C++ and the Output is Verified.

DAY-3 (Data-Structures) Lab Session

EXP - 6

TITLE : C Program for Evaluating Postfix Notation

PROGRAM :

```
#include <stdio.h>
#include <string.h>
#include <stdlib.h>
int main()
{
        char a[100];
        float p[100];
        int i,j,t=-1;
        float v1,v2,k;
        printf("Enter The Postfix Expression:");
        scanf("%s",a);
        int l=strlen(a);
        a[l]='#';
        for(i=0;i<=l;i++)
        {
                if(a[i]>='0' && a[i]<='9')
                {
                        k=a[i]-'0';
                        t+=1;
                        p[t]=k;
                }
                else if(a[i]=='+' || a[i]=='*' || a[i]=='-' || a[i]=='/')
                {
                        if(t>=1){
                        v1=p[t];
                        t--;
                        v2=p[t];
                        t--;
                        switch(a[i])
                        {
                                case '+':
                                        t+=1;
```

```
                        p[t]=v1+v2;
                        break;
                case '-':
                        t+=1;
                        p[t]=v2-v1;
                        break;
                case '*':
                        t+=1;
                        p[t]=v2*v1;
                        break;
                case '/':
                        t+=1;
                        p[t]=v2/v1;
                        break;
                default :
                        printf("Enter The Valid Operation");
            }}
            else
            {
                printf("Invalid Expression");
                break;
            }
        }
        else if(a[i]=='#')
        {
            if(t>0)
            {
                printf("Enter Valid Expression");
            }
            else
            {
                printf("%.2f",p[t]);
            }
        }
    }
}                                   else
```

```
                    printf("%.2f",p[t]);
                }
            }
        }
}
```

INPUT AND OUTPUT :

```
Enter The Postfix Expression:234*+
14.00
```

RESULT :

The C Program for Evaluating Postfix Notation using Stack is Compiled and Executed Using Dev-C++ and the Output is Verified.

DAY-3 (Data-Structures) Lab Session

EXP - 7

TITLE : C Program for Performing Tree traversals

PROGRAM :

```
#include <stdio.h>
#include <stdlib.h>
struct node
{
	int data;
	struct node *right,*left;
};
struct node *root;
void inorder(struct node *root)
{
	if(root==NULL)
	{
		return;
	}
	else
	{
		inorder(root->left);
		printf("%d",root->data);
		inorder(root->right);
	}
}
void preorder(struct node *root)
{
	if(root==NULL)
	{
		return;
	}
	else
	{
		printf("%d",root->data);
		preorder(root->left);
		preorder(root->right);}}
```

```
void postorder(struct node *root)
{
	if(root==NULL)
	{
		return;
	}
	else
	{
		postorder(root->left);
		postorder(root->right);
		printf("%d",root->data);
	}
}
struct node *insert(struct node *root,int e)
{
	if(root==NULL)
	{
		struct node *t;
		t=(struct node*)malloc(sizeof(struct node));
		t->data=e;
		t->right=NULL;
		t->left=NULL;
		return t;
	}
	else if(e<root->data)
	{
		root->left=insert(root->left,e);
	}
	else if(e>root->data)
	{
		root->right=insert(root->right,e);
	}
	return root;
}
```

```
int main()
{
        root=insert(root,3);
        root=insert(root,4);
        root=insert(root,2);
        printf("INORDER:");
        inorder(root);
        printf("\nPREORDER:");
        preorder(root);
        printf("\nPOSTORDER:");
        postorder(root);
}
```

INPUT AND OUTPUT :

```
INORDER:234
PREORDER:324
POSTORDER:243
```

RESULT :

The C Program for Performing Tree Traverses is Compiled and Executed Using Dev-C++ and the Output is Verified.