DATA STRUCTURES LAB PROGRAMS

Part A

1. **Program to sort the given list using selection sort technique.**

```
#include<stdio.h>
void main()
{
        int a[25],i,j,n,k,t;
        printf("Enter the array size");
        scanf("%d",&n);
        printf("\nEnter %d elements:\n",n);
        for(i=0;i<n;i++)
        scanf("%d",&a[i]);
        for(i=0;i<n;i++)
        {
                k=i;
                for(j=i+1;j<n;j++)
                {
                        if(a[k]>a[j])
                        k=j;
                }
                t=a[i];
                a[i]=a[k];
                a[k]=t;
        }
        printf("\nSorted array :\n");
        for(i=0;i<n;i++)
        printf("%d\t",a[i]);
}
```

2. **Program to sort the given list using insertion sort technique.**

```
#include<stdio.h>
int main()
{
   int i,j,n,temp,a[100];
  // clrscr();
   printf("Enter the total number of elements: ");
   scanf("%d",&n);
   printf("Enter the elements: ");
   for(i=0;i<n;i++)
   scanf("%d",&a[i]);
   for(i=1;i<n;i++)
   {
     temp=a[i];
     j=i-1;
     while(j>=0)&&temp<a[j])
    {
```

```
        a[j+1]=a[j];
        j=j-1;
      }
      a[j+1]=temp;
    }
    printf("Sorted elements are: ");
    for(i=0;i<n;i++)
    printf("%d\t",a[i]);
    getch();
    return 0;
}
```

3. **Program to solve Tower of Hanoi using Recursion**

```
#include<stdio.h>
void TOWER(int,char[],char[],char[]);
void main()
{
    int N;
     printf("Enter the number of disk to be transferred: ");
    scanf("%d",&N);
    if(N<1)
    {
       printf("\nIncorrect value");
      //exit(0);
    }
    else
          {
       printf("\nThe following moves are required for N=%d\n\n",N);
       TOWER(N,"BEG","AUX","END");
    }  getch();
 }
void TOWER(int NUM,char A[5],char B[5],char C[5])
{
    if(NUM==1)
    {
       printf("%s-->%s\t",A,C);
       return;
    }
    TOWER(NUM-1,A,C,B);
    printf("%s-->%s\t",A,C);
    TOWER(NUM-1,B,A,C);
    return;
}
```

4. **Program to search an element using recursive binary search technique.**

```
#include<stdio.h>
int Bsearch(int a[],int LB,int UB,int ele);
int main()
```

```
{
 int i,n,a[100],ele,index;
// clrscr();
 printf("Enter the array size: ");
 scanf("%d",&n);
 printf("Enter the array elements: ");
 for(i=0;i<n;i++)
  scanf("%d",&a[i]);
 printf("Enter the elements to be searched: ");
 scanf("%d",&ele);
 index=Bsearch(a,0,n,ele);
 if(index==-1)
  printf("Element is not found in the array");
 else
  printf("Element found at position: %d",index+1);
 getch();
 return 0;
 }
int Bsearch(int a[],int LB,int UB,int ele)
{
 while(LB<=UB)
 {
  int middle=(LB+UB)/2;
  if(a[middle]==ele)
   return middle;
  else if(ele<a[middle])
   return Bsearch(a,LB,middle-1,ele);
  else if(ele>a[middle])
   return Bsearch(a,middle+1,UB,ele);
 }
 return -1;
}
```

5. **Program to implement Stack operations using arrays.**

```
#include<stdio.h>
#define MAX 20
int stack[MAX];
int top=-1;
void push(int ele)
{
        top=top+1;
        stack[top]=ele;
}
int pop()
{
        int ele=stack[top];
        top--;
        return ele;
}
```

```
int is_stackfull()
{
	if (top==MAX-1)
	{
		printf("Stack is full");
		return 1;
	}
	else
	return 0;
}
int is_stackempty()
{
	if (top==-1)
	{
		printf("Stack is empty");
		return 1;
	}
	else
	return 0;
}
void display()
{
	int i;
	for(i=0;i<=top;i++)
	printf("%d\t",stack[i]);
}
void main()
{
	int ch=0;
	while(ch<=4)
	{
		printf("\nMenu\n 1. Push \n 2. Pop\n 3. Display\n 4. Exit\n Enter your
choice:");
		scanf("%d",&ch);
		if(ch==1)
		{
			int e;
			if(is_stackfull())
			break;
			printf("Enter the element to be pushed: ");
			scanf("%d",&e);
			push(e);
			printf("Element is pushed");
		}
		if(ch==2)
		{
			if(is_stackempty())
			break;
			printf("The popped element is %d",pop());
		}
		if(ch==3)
```

```
                {
                        if(is_stackempty())
                        break;
                        display();
                }
                if(ch==4)
                {
                        printf("Exited");
                        break;
                }
        }
}
```

6. **Program to implement Queue operations using arrays**

```
#include<stdio.h>
#define MAX 10
int queue[MAX], front = -1, rear = -1;
void enqueue(int item)
{
   if(rear == MAX-1)
        {
      printf("Queue is full\n");
   }
   else
        {
      if(front == -1)
      {
         front = 0;
         }
      rear = rear + 1;
      queue[rear] = item;
      printf("We have enqueued %d\n",item);
  }
}
void dequeue()
{
   if(front == -1)
        {
      printf("Queue is empty\n");
   }
   else
        {
      printf("We have dequeued : %d\n", queue[front]);
      front = front + 1;
      if(front > rear)
                {
            front = -1;
            rear = -1;
      }
  }
}
void display()
```

```
{
   if(rear == -1)
      printf("\nUnable to display as queue is empty");
   else{
      int i;
      printf("\nThe queue after enqueue & dequeue operations :");

      for(i = front; i <= rear; i++)
                printf("%d ",queue[i]);
  }
}
void main()
{
   int ch;
   while(1)
  {
                    printf("\nMenu\n1. Insert an element into queue\n");
                    printf("2. Delete an element from queue\n");
                    printf("3. Display all elements of queue\n");
                    printf("4. Exit\n");
                    printf("Enter your choice: ");
                    scanf("%d",&ch);
                     if(ch==1)
                    {
                              int e;
                              printf("Enter the element to be inserted: ");
                              scanf("%d",&e);
                              enqueue(e);
                    }
                    if(ch==2)
                    {
                              dequeue();
                    }
                    if(ch==3)
                    {
                              display();
                    }
                    if(ch==4)
                    {
                              printf("Exited");
                              break;
                    }
   }
}
```

Part B

1. **Program to sort the given list using merge sort technique.**

```
#include<conio.h>
#include<stdio.h>
void merge (int a[],int low,int mid,int high)
{
  int temp[100],i,j,k,s;
  i=low;
  j=mid+1;
  k=low;
while(i<=mid&& j<=high)
{
        if (a[i]<a[j])
        {
                temp[k]=a[i];
                i++;
        }
        else
        {
                temp[k]=a[j];
                j++;
        }
        k++;
}
if(i>mid)
{
        for(s=j;s<=high;s++)
        {
                temp[k]=a[s];
                k++;
        }
}
else
{
        for(s=i;s<=mid;s++)
        {
                temp[k]=a[s];
                k++;
        }
}
for(k=low;k<=high;k++)
{
        a[k]=temp[k];
}
}
void mergesort (int a[],int low,int high)
{
```

```
        int mid;
        if(low<high)
        {
                mid=(low+high)/2;
                mergesort(a,low,mid);
                mergesort(a,mid+1,high);
                merge(a,low,mid,high);
        }
}
void main()
{
        int a[100],i,n,low,high;
        printf("\n merge sort\n");
        printf("enter the number of elements ");
        scanf("%d",&n);
        printf("enter the elements :");
        for(i=1;i<=n;i++)
        {
                scanf("%d",&a[i]);
        }
        low=1;
        high=n;
        mergesort(a,low,high);
        printf("\n after sorting:\n");
        for(i=1;i<=n;i++)
        {
                printf("%d \n",a[i]);
        }
}
```

2. **Program to implement circular queue using array.**

```
#include <stdio.h>
#define size 5
int items[size];
int front = 0, rear = -1,count=0;

void enQueue(int ele)
{
  if (count==size)
    printf("\n Queue is full!! \n");
  else
  {
    rear = (rear + 1) % size;
    items[rear] = ele;
    count++;
    printf("\n Inserted -> %d", ele);
  }
}
```

```
void deQueue()
{
  int element;
  if (count==0)
          printf("Queue is underflow\n");
    else
          {
                  printf("\n Deleted element -> %d \n", items[front]);
     front = (front + 1) % size;
    count--;
          }
}
void display()
{
  int i;
  if (count==0)
    printf("\n Empty Queue\n");
  else
  {
    printf("\n Front -> %d ", front);
    printf("\n Items -> ");
    for (i = front; i != rear; i = (i + 1) % size)
           {
      printf("%d ", items[i]);
    }
    printf("%d ", items[i]);
    printf("\n Rear -> %d \n", rear);
}
}
int main()
{
    int choice, data;

    while (1) {
        printf("\nCircular Queue Menu\n");
        printf("1. Enqueue\n");
        printf("2. Dequeue\n");
        printf("3. Display\n");
        printf("4. Exit\n");
        printf("Enter your choice: ");
        scanf("%d", &choice);

        switch (choice) {
            case 1:
                printf("Enter data to enqueue: ");
                scanf("%d", &data);
                enQueue(data);
```

```
            break;
        case 2:
            deQueue();
            break;
        case 3:
            display();
            break;
        case 4:
            printf("Exiting the program.\n");
            return 0;
        default:
            printf("Invalid choice. Please try again.\n");
      }
    }

    return 0;
}
```

3. **Program to implement Stack operations using linked list.**

```
#include<stdio.h>
#include<stdlib.h>
void push(int);
void pop();
void display();
struct node
{
        int info;
        struct node *link;
};
struct node *top=NULL,*new,*temp;
void main()
{
        int option=0,num;
        while(option<4)
        {
        printf("\n select from the following   option\n1.Push\n2.Pop\n3.Display\n4.Exit\nenter
the one of the choice:");
                scanf("%d",&option);
        switch(option)
        {
                case 1:
                                printf("\n enter the item to be pushed:");
                                scanf("%d",&num);
                                push(num);
                                break;

                case 2:
                                pop();
```

```
                                        break;
                        case 3:
                                        display();
                                        break;
                        case 4:
                                        printf("Exit");
                        default:
                                        printf("\n kindly enter the valid option:");
                }
                }
}
void push(int item)
{
                struct node *new=(struct node *)malloc(sizeof(struct node));
                new->info=item;
                new->link=NULL;
                if(top==NULL)
                top=new;
                else
                 {
                        new->link=top;
                        top=new;
                }
}
void pop()
{
                if(top==NULL)
                printf("stack underflow!");
                else
                {
                        temp=top;
                        top=top->link;
                        printf("Element deleted:%d",temp->info);
                        free(temp);
                }
}
void display()
{
                temp=top;
                if(temp==NULL)
                printf("the stack is empty \n");
                else
                {
                        printf("the stack elements are as follows:\n");
                        while(temp!=NULL)
                        {
                                printf("%d\n",temp->info);
```

```
                temp=temp->link;
                }
        }
}
```

4. **Program to evaluate postfix expression.**

```
#include<stdio.h>
#include<math.h>
#include<string.h>
int stack[20];
int top=0;
void push(int x)
{
        stack[++top]=x;
}
int pop()
{
        return stack[top--];
}
void main()
{
        char post[20],symb;
        int a,b,value,i,len,res=0;
        printf("Enter the expression\n(Single digit number and operator dont use space
between symbols):\n");
        scanf("%s",&post);
        len=strlen(post);
        for(i=0;i<len;i++)
        {
                symb=post[i];
         switch(symb)
          {
                case '+':       a=pop();
                                b=pop();
                                value=b+a;
                                push(value);
                                break;
                case '-':       a=pop();
                                b=pop();
                                value=b-a;
                                push(value);
                                break;
                case '*':       a=pop();
                                b=pop();
                                value=b*a;
                                push(value);
                                break;
                case '/':       a=pop();
```

```
                                        b=pop();
                                        value=b/a;
                                        push(value);
                                        break;
                case '^':       a=pop();
                                        b=pop();
                                        value=pow(b,a);
                                        push(value);
                                        break;
                case '%':       a=pop();
                                        b=pop();
                                        value=b%a;
                                        push(value);
                                        break;
                default: push(symb-48);
        }
 }
res=pop();
printf("\n The result of expression %s = %d\n\n",post,res);
}
```

5. **Program to perform insert node at the end, delete a given node and display contents of single linked list.**

```
#include<stdio.h>
#include<stdlib.h>
void insertAtEnd();
void display();
void Remove();

struct Node
{
        int info;
        struct Node *link;
}*first,*save,*temp,*New,*p;

void insertAtEnd()
{
        New =malloc(sizeof(struct Node));
        printf("\nEnter the infomation for the node:");
        scanf("%d",&New ->info);
        New ->link= NULL;
        if(first == NULL)
                first = New ;
        else
        {
                save = first;
                while(save->link != NULL)
                save = save->link;
```

```
                save->link = New;
}
printf("One node inserted!!!\n");
}
void Remove()
{
        int item;
        printf("\nEnter information to be deleted:");
        scanf("%d",&item);
        if(first==NULL)
        {
                printf("list is empty");
                return;
        }
        else if(item==first->info)
        {
                save=first;
                first=first->link;
                free(save);
                return;
                }
        p=NULL;
        save=first;
        while(save!=NULL &&item!=save->info)
        {
                p=save;
                save=save->link;
        }
        if(save==NULL)
        {
                printf("item not found");
                return;
        }
        p->link=save->link;
        free(save);
        return;
}
void display()
{
        if(first == NULL)
        {
                printf("\nList is Empty\n");
                return;
        }
        else
        {
                temp = first;
```

```
        }
        printf("\n\nList elements are - \n");
        while(temp!= NULL)
        {
                printf("%d --->",temp->info);
                temp = temp->link;
        }
}
void main()
{
        int ch;
        do
        {
                printf("\n****** MENU ******\n1. Insert At End\n2. Delete specific node\n3.
Display\n4. Exit\nEnter your choice:");
                scanf("%d",&ch);
                switch(ch)
                {
                        case 1:
                                        insertAtEnd();
                                        break;
                        case 2:
                                        Remove();
                                        break;
                        case 3:
                                        display();
                                        break;
                        case 4:
                                        exit(0);
                        default: printf("\nWrong Input!! Try again!!!\n\n");
                }
        }
        while(ch!=4);
}
```

6. **Menu driven program for the following operations on Binary Search Tree(BST) of Integers**
   (a) Create a BST of N Integers
   (b) Traverse the BST in Inorder, Preorder and Post Order.

```
#include <stdio.h>
#include <stdlib.h>
struct node
{
        int info;
        struct node *left, *right;
```

```
};
typedef struct node *Node;
Node insert(Node root,int x)
{
	Node p= (struct node *)malloc(sizeof(struct node));
	p->info = x;
	p->left = p->right = NULL;
	if(root==NULL)
	{
		root=p;
		return root;
	}
	else if(x>root->info)
		root->right=insert(root->right,x);
	else if(x<root->info)
		root->left=insert(root->left,x);
	else
		printf("\nDuplicate entry");
		return root;
}
void inorder(Node root)
{
	if(root!=NULL)
	{
		inorder(root->left);
		printf("%d\t",root->info);
		inorder(root->right);
	}
}
void preorder(Node root)
{
	if(root!=NULL)
	{
		printf("%d\t",root->info);
		preorder(root->left);
		preorder(root->right);
	}
}
void postorder(Node root)
{
	if(root!=NULL)
	{
		postorder(root->left);
		postorder(root->right);
		printf("%d\t",root->info);
	}
}
```

```
void main()
{
	Node root;
	int ch,ele,n,i;
	root=NULL;
	printf("Menu\n1.InsertNode\n2.Inorder\n3.Preorder\n4.Postorder\n5.Exit\n");
	while(1)
	{
		printf("\nEnter your choice:");
		scanf("%d",&ch);
		switch(ch)
		{
			case 1: printf("\nHow many values?:");
				scanf("\n%d",&n);
				printf("\nEnter %d elements:",n);
				for(i=0;i<n;i++)
				{
					scanf("%d",&ele);
					root=insert(root,ele);
				}
				break;
			case 2: printf("\nElements in Inorder\n");
				inorder(root);
				break;
			case 3: printf("\nElements in Preorder\n");
				preorder(root);
				break;
			case 4: printf("\nElements in postorder\n");
				postorder(root);
				break;
			case 5:exit(0);
		}
	}
}
```