DHARMSINH DESAI UNIVERSITY

MCA SEM -2

DATA STRUCTURE USING C ASSIGNMENT SUBMISSION

TERMWORK SUBMISSION

NAME : GUJARATI POOJAN P.

ROLL NO: MA004

STUDENT ID : 18MAPOG013

E-MAIL : poojangujarati20@gmail.com

SUBMITTED TO,

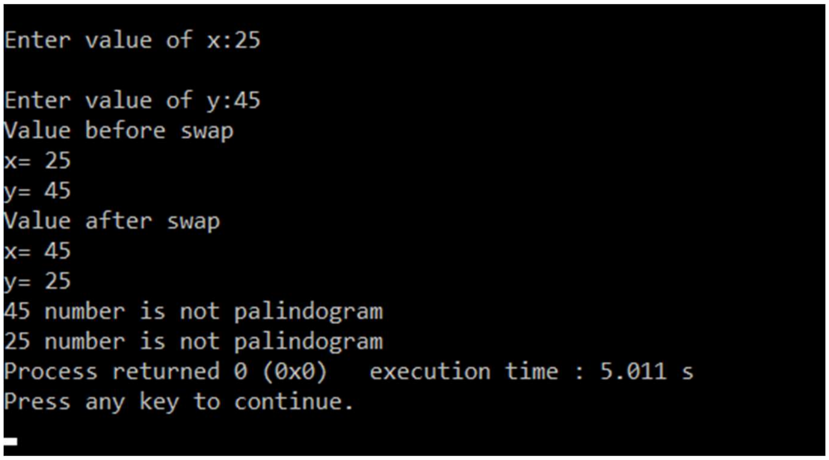
PROF. HIMANSHU PUROHIT

1. Create function called swap ( ), which swaps the number values. Create a function pointer which points to a swap ( ) function and call function using pointer. Write a program which also checks whether the two number entered by user is palindrome or not after swaping.

```
#include<stdio.h>

void  swap(int *a,int *b)

{

        int tmp;

        tmp=*a;

        *a=*b;

        *b=tmp;

}

int reverse(int n)

{

        int rem,rnum=0;

        while(n > 0)

        {

                rem= n % 10;

                rnum=rnum * 10 + rem;

                n= n / 10;

        }

        return rnum;

}

int main()

{

        FILE *fp;

        fp=fopen("prog1.txt","r");
```

```
    int x,y;

    printf("\nEnter value of x:");

    scanf("%d",&x);

    fscanf(fp,"%d",&x);

    printf("\nEnter value of y:");

    scanf("%d",&y);

    fscanf(fp,"%d",&y);

    printf("Value before swap \nx= %d \ny= %d",x,y);



    void (*pfun) (int *,int *)=swap;

    pfun(&x,&y);

    printf("\nValue after swap \nx= %d \ny= %d",x,y);


    int (*rp) (int )=reverse;

    if(x == rp(x))

        printf("\n%d number is palindogram",x);

    else

        printf("\n%d number is not palindogram",x);


    if(y == rp(y))

        printf("\n%d number is palindogram",y);

    else

        printf("\n%d number is not palindogram",y);

    fclose(fp);


}
```

Output:



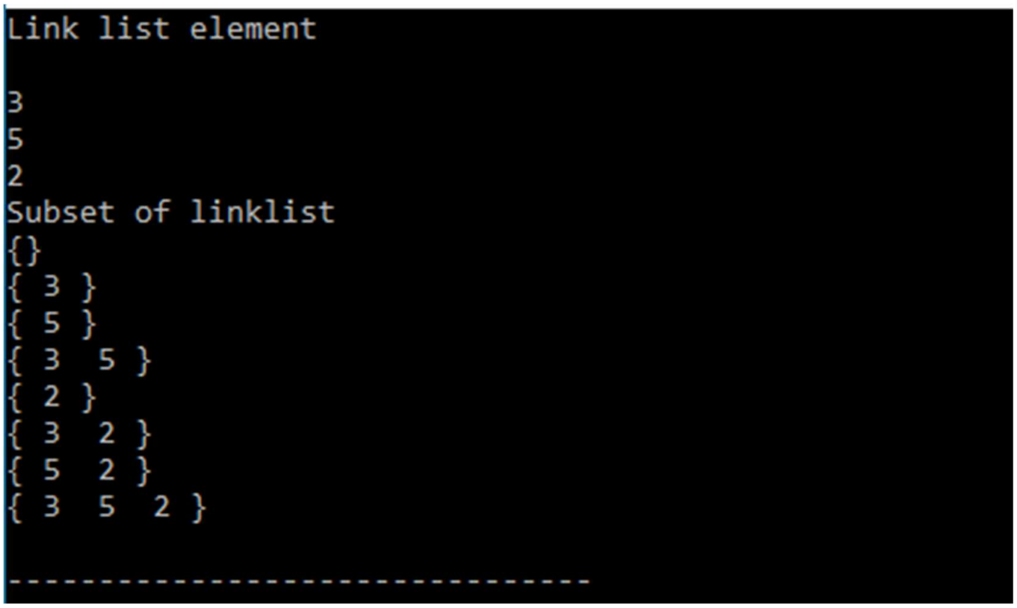
```
Enter value of x:25

Enter value of y:45
Value before swap
x= 25
y= 45
Value after swap
x= 45
y= 25
45 number is not palindogram
25 number is not palindogram
Process returned 0 (0x0)   execution time : 5.011 s
Press any key to continue.
```

2. **Implement linked list to create and manage a set of elements. Set of elements contains integer values i.e. S = {4,5,6}. Also implement a method which shows all possible subsets of the created set by user i.e. {{4}, {5}, {6}, {4,5}, {4,6}, {5,6}, {4,5,6}, {Ø}}.**

```
#include<stdio.h>

void  swap(int *a,int *b)

{

    int tmp;

    tmp=*a;

    *a=*b;

    *b=tmp;

}

int reverse(int n)

{

    int rem,rnum=0;

    while(n > 0)

    {

            rem= n % 10;

            rnum=rnum * 10 + rem;

            n= n / 10;
    }
    return rnum;
}
int main()
{
    FILE *fp;
    fp=fopen("prog1.txt","r");
    int x,y;
    printf("\nEnter value of x:");
    scanf("%d",&x);
    fscanf(fp,"%d",&x);
    printf("\nEnter value of y:");
    scanf("%d",&y);
    fscanf(fp,"%d",&y);
    printf("Value before swap \nx= %d \ny= %d",x,y);



    void (*pfun) (int *,int *)=swap;
    pfun(&x,&y);
    printf("\nValue after swap \nx= %d \ny= %d",x,y);

    int (*rp) (int )=reverse;
    if(x == rp(x))
            printf("\n%d number is palindogram",x);
    else
            printf("\n%d number is not palindogram",x);

    if(y == rp(y))
            printf("\n%d number is palindogram",y);
```

```
        else
                printf("\n%d number is not palindogram",y);
        fclose(fp);

    }
```

Output:

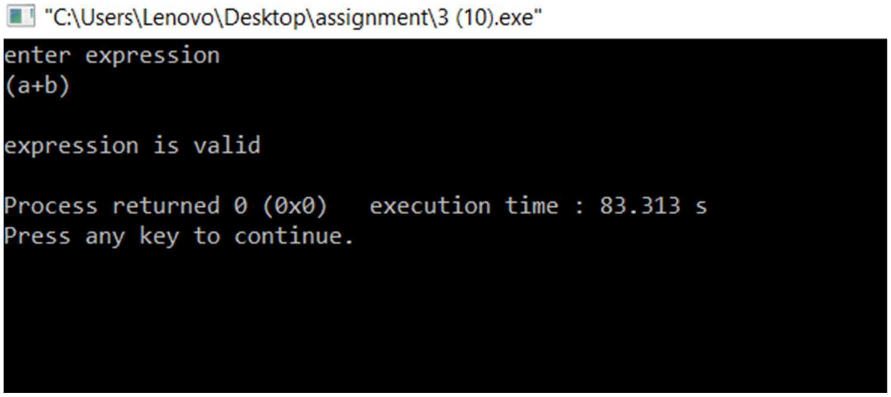


```
Link list element

3
5
2
Subset of linklist
{}
{ 3 }
{ 5 }
{ 3  5 }
{ 2 }
{ 3  2 }
{ 5  2 }
{ 3  5  2 }

--------------------------------
```

3. **Write a program to check the balance of parenthesis if an expression. Implement required data structure for the same**

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

int top = -1;
char stack[100];
void push(char);
void pop();
void find_top();

void main()
{
        int i;
        char a[100];
        printf("enter expression\n");
        scanf("%s", &a);
        for (i = 0; a[i] != '\0';i++)
        {
                if (a[i] == '(')
                {
                        push(a[i]);
                }
                else if (a[i] == ')')
                {
                        pop();
                }
        }
        find_top();
}
void push(char a)
{
        stack[top] = a;
        top++;
}
void pop()
{
        top--;
}
void find_top()
{
        FILE *ptr;
        ptr=fopen("expression.txt","w");
        if (top == -1)
        {
                printf("\nexpression is valid\n",top);
                fprintf(ptr,"expression is valid");
        }
        else
        {
                printf("\nexpression is invalid\n");
                fprintf(ptr,"expression is invalid");
        }
        fclose(ptr);
}
```

Output:



"C:\Users\Lenovo\Desktop\assignment\3 (10).exe"

```
enter expression
(a+b)

expression is valid

Process returned 0 (0x0)   execution time : 83.313 s
Press any key to continue.
```

4. **Implement a program to generate a linked list. For any unsorted linked list, write a method that will delete any duplicates from the linked list without using a temporary buffer.**

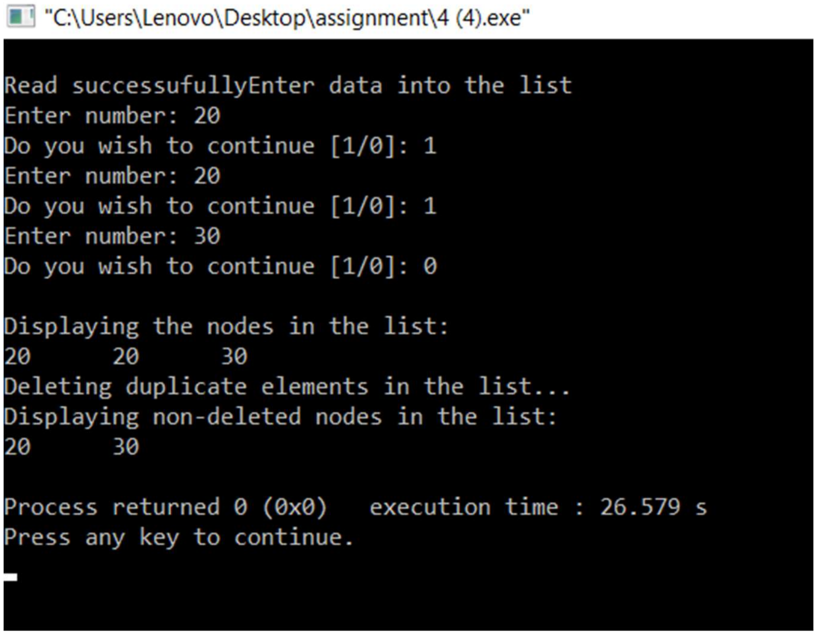
```
void write(struct node *);
int main()
{
   struct node *p = NULL;
   struct node_occur *head = NULL;
     read(&p);
   int n;
   printf("Enter data into the list\n");
   create(&p);
   write(&p);
   printf("Displaying the nodes in the list:\n");
   display(p);
   printf("Deleting duplicate elements in the list...\n");
   dup_delete(&p);
   printf("Displaying non-deleted nodes in the list:\n");
   display(p);
   release(&p);

   return 0;
}

void dup_delete(struct node **head)
{
   struct node *p, *q, *prev, *temp;

   p = q = prev = *head;
```

```
    q = q->next;
    while (p != NULL)
    {
        while (q != NULL && q->num != p->num)
        {
            prev = q;
            q = q->next;
        }
        if(q == NULL)
        {
            p = p->next;
            if (p != NULL)
            {
                q = p->next;
            }
        }
        else if (q->num == p->num)
        {
            prev->next = q->next;
            temp = q;
            q = q->next;
            free(temp);
        }
    }
}

void create(struct node **head)
{
    int c, ch;
    struct node *temp, *rear;

    do
    {
        printf("Enter number: ");
        scanf("%d", &c);
        temp = (struct node *)malloc(sizeof(struct node));
        temp->num = c;
        temp->next = NULL;
        if (*head == NULL)
        {
            *head = temp;
        }
        else
        {
            rear->next = temp;
        }
        rear = temp;
        printf("Do you wish to continue [1/0]: ");
        scanf("%d", &ch);
    } while (ch != 0);
    printf("\n");
}

void display(struct node *p)
{
    while (p != NULL)
    {
        printf("%d\t", p->num);
```

```
      p = p->next;
   }
   printf("\n");
}
void write(struct node *p)
{
     int data,i;
     FILE *ptr;
     ptr=fopen("sortlink.txt","w");
     while (p != NULL)
   {
      fprintf(ptr,"%d\t", p->num);
      p = p->next;
   }
     fclose(ptr);
}
void read(struct node *p)
{
     int i;
     FILE *fptr;
     fptr=fopen("sortlink.txt","r");
     while (p != NULL)
   {
      fscanf(fptr,"%d\t", &p->num);
      p = p->next;
   }
     fclose(fptr);
     printf("\nRead successufully");
}
```

```
void release(struct node **head)

{

    struct node *temp = *head;

    *head = (*head)->next;

    while ((*head) != NULL)

    {

        free(temp);

        temp = *head;

        (*head) = (*head)->next;

    }

}
```

Output:



"C:\Users\Lenovo\Desktop\assignment\4 (4).exe"

```
Read successufullyEnter data into the list
Enter number: 20
Do you wish to continue [1/0]: 1
Enter number: 20
Do you wish to continue [1/0]: 1
Enter number: 30
Do you wish to continue [1/0]: 0

Displaying the nodes in the list:
20      20      30
Deleting duplicate elements in the list...
Displaying non-deleted nodes in the list:
20      30

Process returned 0 (0x0)   execution time : 26.579 s
Press any key to continue.
```

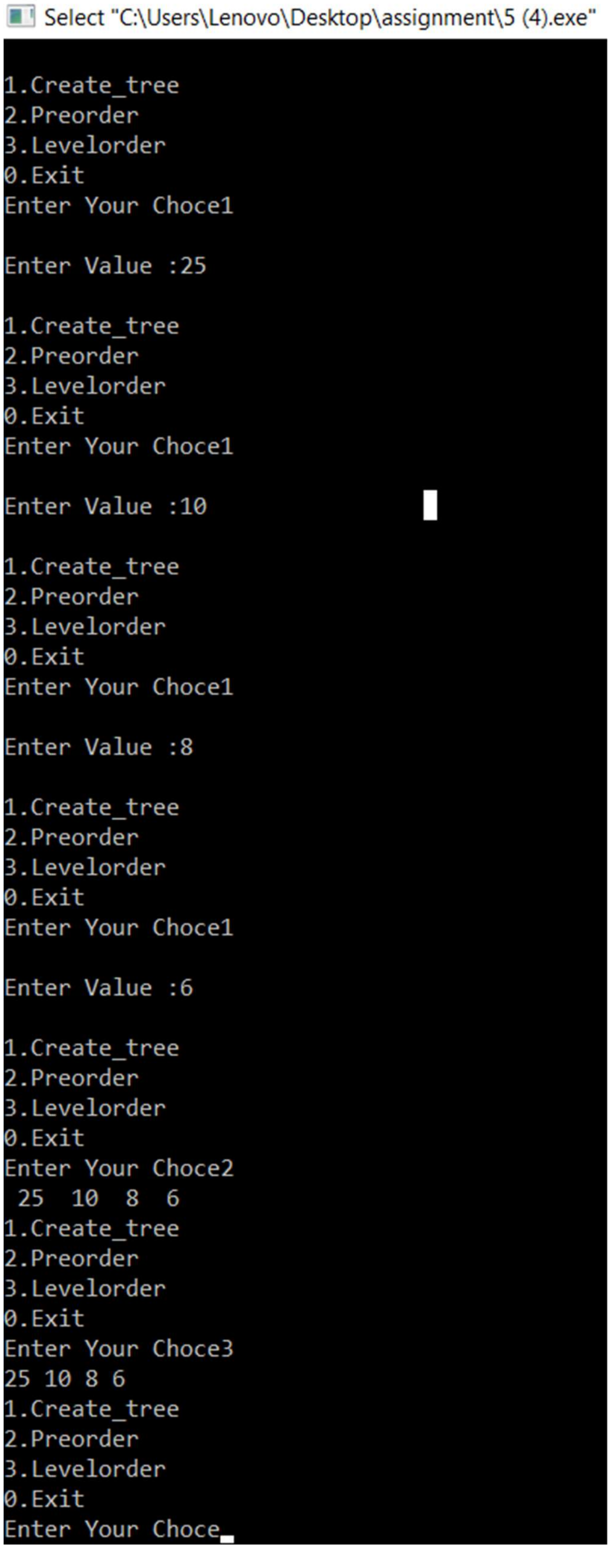
5. **Write a program to create a binary tree. Implement required method to generate a binary tree from user inputs and to display binary tree using level order and pre order traversals.**

```
#include<stdio.h>

#include<stdlib.h>

struct tree

{

        int data;

        struct tree *left;

        struct tree *right;

};

struct tree *root=NULL;

struct tree *create_tree(struct tree *info,int no)

{

        if(info==NULL)

        {

                info=(struct tree *)malloc(sizeof(struct tree));

                info->data=no;

                info->left=NULL;

                info->right=NULL;

        }

        else

        {

                if(no<=info->data)

                {

                        info->left=create_tree(info->left,no);

                }

                else

                {

                        info->right=create_tree(info->right,no);

                }
```

```
		}
}
void pre_order(struct tree *info)
{
		if(info!=NULL)
		{
			printf(" %d ",info->data);
			pre_order(info->left);
			pre_order(info->right);
		}
}
void level_order(struct tree* info)
{
  int h = height(info);
  int i;
  for (i=1; i<=h; i++)
    printGivenLevel(info, i);
}
void printGivenLevel(struct tree* info, int level)
{
  if (info == NULL)
    return;
  if (level == 1)
    printf("%d ", info->data);
  else if (level > 1)
  {
    printGivenLevel(info->left, level-1);
    printGivenLevel(info->right, level-1);
  }
```

```
}
int height(struct tree* info)
{
  if (info==NULL)
    return 0;
  else
  {
    int lheight = height(info->left);
    int rheight = height(info->right);
    if (lheight > rheight)
      return(lheight+1);
    else
                return(rheight+1);
  }
}
int main()
{
        int ch,n;
        do
        {
                printf("\n1.Create_tree");
                printf("\n2.Preorder");
                printf("\n3.Levelorder");
                printf("\n0.Exit");
                printf("\nEnter Your Choce");
                scanf("%d",&ch);
                switch(ch)
                {
                        case 1 :printf("\nEnter Value :");
                scanf("%d",&n);

                root=create_tree(root,n);

                break;

            case 2 :pre_order(root);

                break;

            case 3 :level_order(root);

                break;

            case 0 :exit(0);

                break;

            default:printf("\nInvalid Choice");

                break;

        }

    }while(ch!=0);

    return 0;

}
```

Output:



Select "C:\Users\Lenovo\Desktop\assignment\5 (4).exe"

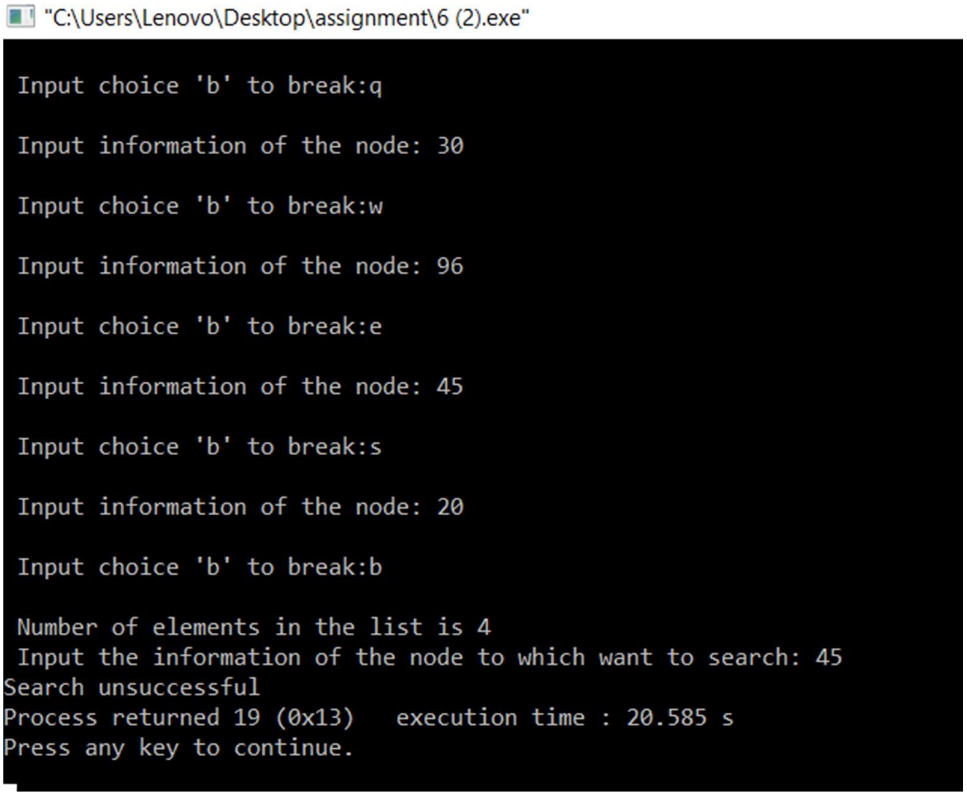
```

1.Create_tree
2.Preorder
3.Levelorder
0.Exit
Enter Your Choce1

Enter Value :25

1.Create_tree
2.Preorder
3.Levelorder
0.Exit
Enter Your Choce1

Enter Value :10

1.Create_tree
2.Preorder
3.Levelorder
0.Exit
Enter Your Choce1

Enter Value :8

1.Create_tree
2.Preorder
3.Levelorder
0.Exit
Enter Your Choce1

Enter Value :6

1.Create_tree
2.Preorder
3.Levelorder
0.Exit
Enter Your Choce2
 25  10  8  6
1.Create_tree
2.Preorder
3.Levelorder
0.Exit
Enter Your Choce3
25 10 8 6
1.Create_tree
2.Preorder
3.Levelorder
0.Exit
Enter Your Choce_
```

6. **Given two values v1 and v2 (where v1 < v2) within a Binary Search Tree. Print all the keys of tree in range v1 to v2. i.e. print all x such that v1<=x<=v2 and x is a element of given BST. (Create a Binary Search Tree by any method).**

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

struct NODE
{
    char Info;
    struct NODE *Left_Child;
    struct NODE *Right_Child;
};

int flag = 0;
struct NODE *Binary_Tree (char *, int, int);
int Search_Node(struct NODE *, char);
struct NODE *  Binary_Tree (char *List, int Lower, int Upper)
{
    struct NODE *Node;
    int Mid = (Lower + Upper)/2;
    Node = (struct NODE*) malloc(sizeof(struct NODE));

    Node->Info = List [Mid];
    if ( Lower>= Upper)
    {
        Node->Left_Child = NULL;
        Node->Right_Child = NULL;
        return (Node);
    }
```

```
        if (Lower <= Mid - 1)
                Node->Left_Child = Binary_Tree (List, Lower, Mid - 1);
        else
                Node->Left_Child = NULL;
        if (Mid + 1 <= Upper)
                Node->Right_Child = Binary_Tree (List, Mid + 1, Upper);
        else
                Node->Right_Child = NULL;
        return(Node);
}
int Search_Node(struct NODE *Node, char Info)
{
        while (Node != NULL)
        {
                if (Node->Info == Info)
                {
                        flag = 1;
                        return(flag);
                }
                else
                        if(Info < Node->Info)
                        {
                                Node = Node->Left_Child;
                        }
                        else
                        {
                                Node = Node->Right_Child;
                        }
        }
        return(flag);
}
void main()
{
        int flag;
        char List[100];
        int Number = 0;
        char Info ;
        char choice;
        struct NODE *T = (struct NODE *) malloc(sizeof(struct NODE));
        T = NULL;
        printf("\n Input choice 'b' to break:");
        choice = getchar();
        while(choice != 'b')
        {
                fflush(stdin);
                printf("\n Input information of the node: ");
                scanf("%c", &Info);
                List[Number++] = Info;
                fflush(stdin);
                printf("\n Input choice 'b' to break:");
                choice = getchar();
        }
        Number --;
        printf("\n Number of elements in the list is %d", Number+1);
        T = Binary_Tree(List, 0, Number);
//      Output(T, 1);
        fflush(stdin);
        printf("\n Input the information of the node to which want to search: ");
        scanf("%c", &Info);

        flag = Search_Node(T, Info);

        if (flag)

        {

                printf("\n Search is successful \n");

        }

        else

                printf("Search unsuccessful");

}
```

Output:



"C:\Users\Lenovo\Desktop\assignment\6 (2).exe"

```
 Input choice 'b' to break:q

 Input information of the node: 30

 Input choice 'b' to break:w

 Input information of the node: 96

 Input choice 'b' to break:e

 Input information of the node: 45

 Input choice 'b' to break:s

 Input information of the node: 20

 Input choice 'b' to break:b

 Number of elements in the list is 4
 Input the information of the node to which want to search: 45
Search unsuccessful
Process returned 19 (0x13)   execution time : 20.585 s
Press any key to continue.
```

7. **Write a program to create a binary tree. Implement required method to generate a binary tree from user inputs and check whether the Binary Tree is a perfect binary tree.**

```
#include<stdio.h>

#include<stdlib.h>

struct tree

{

    int data;

    struct tree *left;

    struct tree *right;

};

struct tree *root=NULL;

struct tree *create_tree(struct tree *info,int no)

{

    if(info==NULL)

    {

        info=(struct tree *)malloc(sizeof(struct tree));

        info->data=no;

        info->left=NULL;

        info->right=NULL;

    }

    else

    {

        if(no<=info->data)

        {

            info->left=create_tree(info->left,no);

        }

        else

        {

            info->right=create_tree(info->right,no);

        }

    }
}
void pre_order(struct tree *info)
{
        if(info!=NULL)
        {
                printf(" %d ",info->data);
                pre_order(info->left);
                pre_order(info->right);
        }
}
void post_order(struct tree *info)
{
        if(info!=NULL)
        {
                post_order(info->left);
                post_order(info->right);
                printf(" %d ",info->data);
        }
}
void in_order(struct tree *info)
{
        if(info!=NULL)
        {
                in_order(info->left);
                printf(" %d ",info->data);
                in_order(info->right);
        }
}
int isfulltree(struct tree *info)
```

```
{
        if(info==NULL)
                return 1;
        if(info->left == NULL && info->right == NULL)
                return 1;
        if((info->left) && (info->right))
                return (isfulltree(info->left)&&isfulltree(info->right));
}
int main()
{
        int ch,n,l;
        do
        {
                printf("\n1.Create_tree");
                printf("\n2.Inorder");
                printf("\n3.Preorder");
                printf("\n4.Postorder");
                printf("\n5.Prefect Tree or not");
                printf("\n6.Exit");
                printf("\nEnter Your Choce");
                scanf("%d",&ch);
                switch(ch)
                {
                        case 1 :printf("\nEnter Value :");
                                scanf("%d",&n);
                                root=create_tree(root,n);
                                break;
                        case 2 :in_order(root);
                                break;
```

```
            case 3 :pre_order(root);
                break;
            case 4 :post_order(root);
                break;
            case 5:if (isfulltree(root))
                        printf("\nTree is perfect:");
                    else
                        printf("\nTree is not perfect");
                    break;
            case 6 :exit(0);
                break;
            default:printf("\nInvalid Choice");
                break;
        }
    }while(ch!=6);
    return 0;
}
```

Output:

```
1.Create_tree
2.Inorder
3.Preorder
4.Postorder
5.Prefect Tree or not
6.Exit
Enter Your Choce1

Enter Value :20
```

```
1.Create_tree
2.Inorder
3.Preorder
4.Postorder
5.Prefect Tree or not
6.Exit
Enter Your Choce1

Enter Value :40

1.Create_tree
2.Inorder
3.Preorder
4.Postorder
5.Prefect Tree or not
6.Exit
Enter Your Choce1

Enter Value :65

1.Create_tree
2.Inorder
3.Preorder
4.Postorder
5.Prefect Tree or not
6.Exit
Enter Your Choce1
Enter Value :35


1.Create_tree
2.Inorder
3.Preorder
4.Postorder
5.Prefect Tree or not
6.Exit
Enter Your Choce1


Enter Value :85


1.Create_tree
2.Inorder
3.Preorder
4.Postorder
5.Prefect Tree or not
6.Exit
Enter Your Choce2
 20  35  40  65  85
1.Create_tree
2.Inorder
3.Preorder
4.Postorder
5.Prefect Tree or not
6.Exit
Enter Your Choce3
 20  40  35  65  85
1.Create_tree
```

```
2.Inorder
3.Preorder
4.Postorder
5.Prefect Tree or not
6.Exit
Enter Your Choce4
 35  85  65  40  20
1.Create_tree
2.Inorder
3.Preorder
4.Postorder
5.Prefect Tree or not
6.Exit
Enter Your Choce5

Tree is not perfect
1.Create_tree
2.Inorder
3.Preorder
4.Postorder
5.Prefect Tree or not
6.Exit
Enter Your Choce
```

8. Write a program to implement stack with all basic operations using linked list.

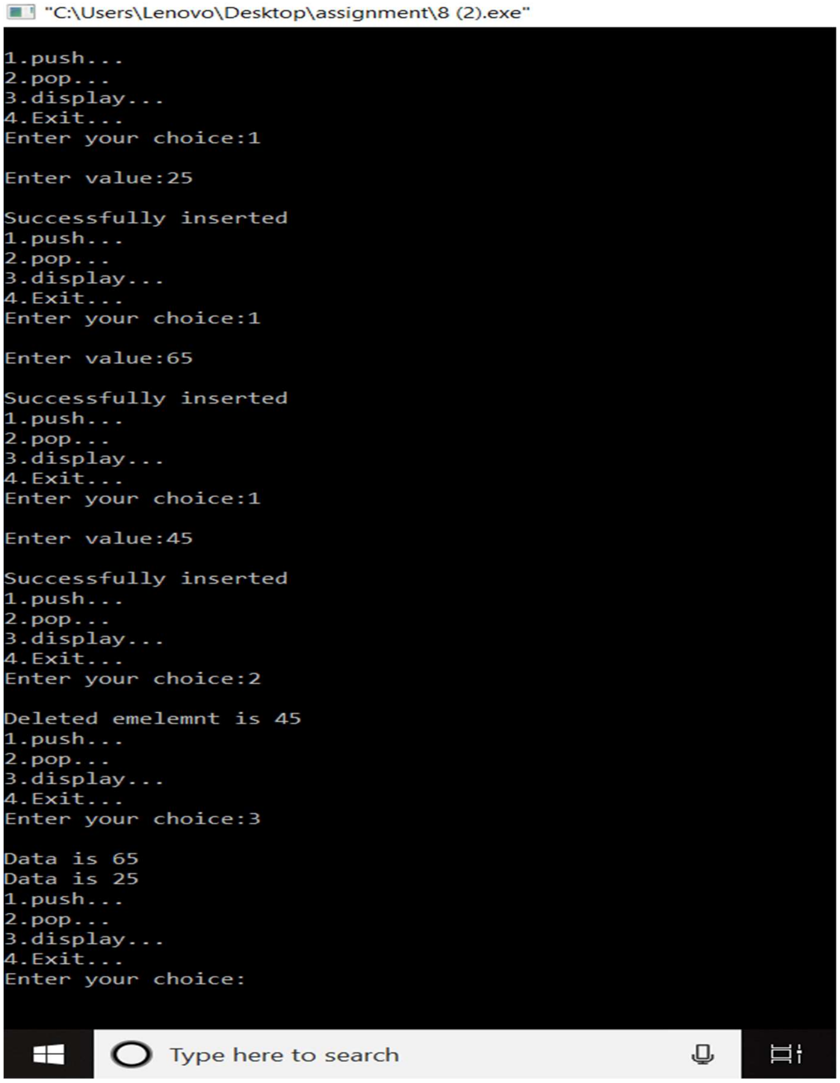
```
#include<stdio.h>
#include<conio.h>
#include<stdlib.h>
```

```
struct node
{
    int data;
    struct node *next;
}*top=NULL;
void push(int);
void pop();
void disp();
void read();
void write();
int main()
{
    read();
    int ch,value;
    do
    {
        printf("\n1.push...");
        printf("\n2.pop...");
        printf("\n3.display...");
        printf("\n4.Exit...");
        printf("\nEnter your choice:");
        scanf("%d",&ch);
        switch(ch)
        {
            case 1: printf("\nEnter value:");
                    scanf("%d",&value);
                    push(value);
                    write();
                    break;
            case 2: pop();
                    write();
                    break;
            case 3: disp();
                    break;
            case 4: exit(1);
                    break;
            default:printf("\nWrong Choice...");
                break;
        }
    }while(ch!=4);

}
void push(int x)
{
    struct node *new_node;
    new_node=(struct node*)malloc(sizeof(struct node));
    new_node->data=x;
    if(top==NULL)
    {
        new_node->next=NULL;
    }
    else
    {
        new_node->next=top;
        //top=new_node;
    }
    top=new_node;
    printf("\nSuccessfully inserted");
}
void pop()
{
    if(top==NULL)
        printf("\nStacklist is not created");
    else
    {
        struct node *temp;
        temp=top;
        printf("\nDeleted emelemnt is %d",temp->data);
        top=temp->next;
        free(temp);
    }
}
void disp()
{
    if(top==NULL)
        printf("\nStack is empty");
    else
    {
        struct node *temp;
        temp=top;
        while(temp!=NULL)
        {
            printf("\nData is %d",temp->data);
            temp=temp->next;
        }
        //printf("\nData is %d",temp->data);
```

```
	}
}
void write()
{
	int data,i;
	FILE *ptr;
	ptr=fopen("stacklink.txt","w");
	struct node *temp;
	temp=top;
	while(temp!=NULL)
	{
		fprintf(ptr,"%d",temp->data);
		temp=temp->next;
	}
	fclose(ptr);
}
void read()
{
	int i;
	FILE *fptr;
	fptr=fopen("stacklink.txt","r");
	struct node *temp;
	temp=top;
	while(temp!=NULL)
	{
		fscanf(fptr,"%d",&temp->data);
		temp=temp->next;
	}
```

```
        fclose(fptr);

}
```

Output:



```
1.push...
2.pop...
3.display...
4.Exit...
Enter your choice:1

Enter value:25

Successfully inserted
1.push...
2.pop...
3.display...
4.Exit...
Enter your choice:1

Enter value:65

Successfully inserted
1.push...
2.pop...
3.display...
4.Exit...
Enter your choice:1

Enter value:45

Successfully inserted
1.push...
2.pop...
3.display...
4.Exit...
Enter your choice:2

Deleted emelemnt is 45
1.push...
2.pop...
3.display...
4.Exit...
Enter your choice:3

Data is 65
Data is 25
1.push...
2.pop...
3.display...
4.Exit...
Enter your choice:
```

9. Write a program to implement Queue with all basic operations using linked list.

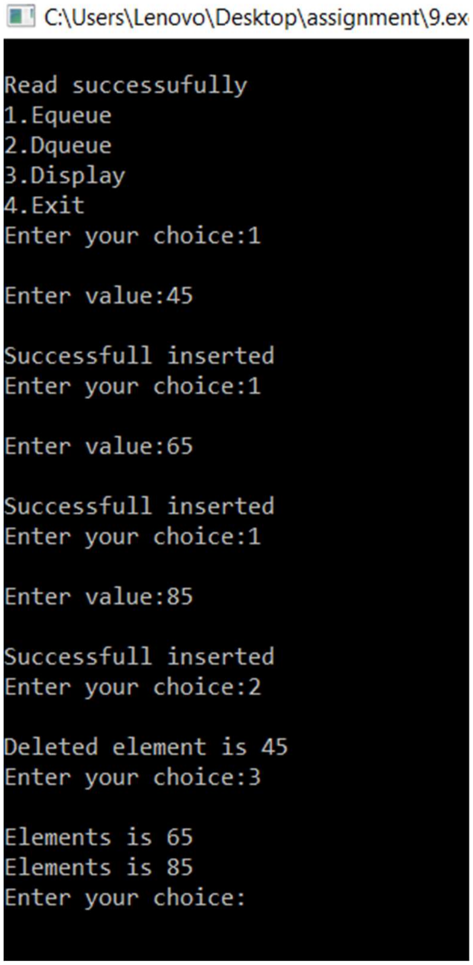
```
#include<stdio.h>

#include<conio.h>

#include<stdlib.h>
```

```
struct node
{
        int data;
        struct node *next;
}*front=NULL,*rear=NULL;
void equeue(int);
void dqueue();
void disp();
void read();
void write();
int main()
{
        read();
        int ch,value;
        printf("\n1.Equeue");
        printf("\n2.Dqueue");
        printf("\n3.Display");
        printf("\n4.Exit");
        do
        {
                printf("\nEnter your choice:");
                scanf("%d",&ch);
                switch(ch)
                {
                        case 1:printf("\nEnter value:");
                                        scanf("%d",&value);
                                        equeue(value);
                                        write();
                                        break;
```

```
            case 2:dqueue();
                    write();
                    break;
            case 3:disp();
                    break;
            default:
                    break;
        }
    }while(ch!=4);
}
void equeue(int x)
{
    struct node *new_node;
    new_node=(struct node*)malloc(sizeof(struct node));
    new_node->data=x;
    new_node->next=NULL;
    if(rear==NULL)
        front=rear=new_node;
    else
    {
        rear->next=new_node;
        rear=new_node;
    }
    printf("\nSuccessfull inserted");
}
void dqueue()
{
    if(front==NULL)
        printf("\nQueue is empty");
    else
    {
        struct node *temp;
        temp=front;
        printf("\nDeleted element is %d",temp->data);
        front=temp->next;
        free(temp);
    }
}
void disp()
{
    if(front==NULL)
        printf("\nQueue is empty");
    else
    {
        struct node *temp;
        temp=front;
        while(temp!=NULL)
        {
            printf("\nElements is %d",temp->data);
            temp=temp->next;
        }
    }
}
void write()
{
    int data,i;
    FILE *ptr;
    ptr=fopen("queuelink.txt","w");
```

```
        struct node *temp;
        temp=front;
        while(temp!=NULL)
        {
                fprintf(ptr,"%d",temp->data);
                temp=temp->next;
        }
        fclose(ptr);
}
void read()
{
        int i;
        FILE *fptr;
        fptr=fopen("queuelink.txt","r");
        struct node *temp;
        temp=front;
        while(temp!=NULL)
        {
                fscanf(fptr,"%d",&temp->data);
                temp=temp->next;
        }
        fclose(fptr);
        printf("\nRead successufully");
}
```

Output:

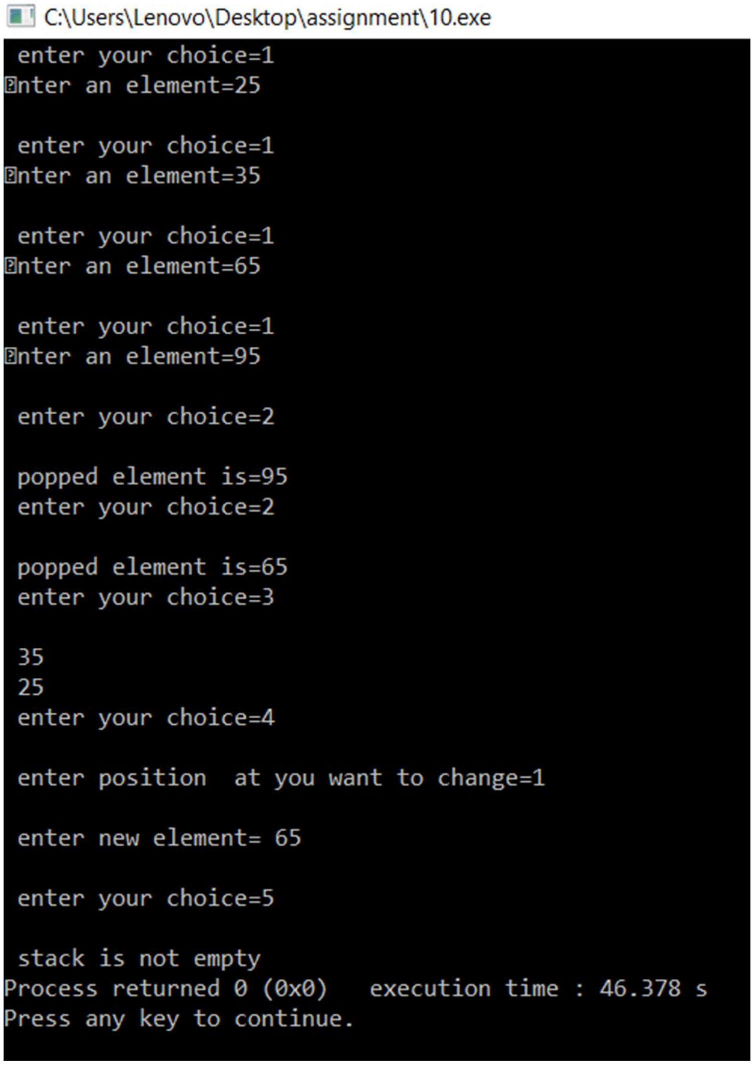


```
Read successufully
1.Equeue
2.Dqueue
3.Display
4.Exit
Enter your choice:1

Enter value:45

Successfull inserted
Enter your choice:1

Enter value:65

Successfull inserted
Enter your choice:1

Enter value:85

Successfull inserted
Enter your choice:2

Deleted element is 45
Enter your choice:3

Elements is 65
Elements is 85
Enter your choice:
```

10. Write a program to implement stack with required operations using array.

```
#include<stdio.h>

#include<conio.h>

#define size 5

int stack[size];

int top=-1;

void push();

void display();

void isEmpty();

void pop();
void update();
int main()
{
            read();
    int ch;
    printf("\n1) push operation");
    printf("\n2) pop operation");
    printf("\n3) peep operation");
    printf("\n4) update operation");
    printf("\n5) isEmpty operation");
    printf("\n6) exit");
    do
    {


    printf("\n enter your choice=");
    scanf("%d",&ch);
    switch(ch)
    {
        case 1:
            push();
            write();
            break;
        case 2:
            pop();
            write();
            break;
        case 3:
            peep();
            break;
        case 4:
            update();
            write();
            break;
        case 5:
            isEmpty();
            break;
        default:
            return 0;
            break;

    }
    }while(ch!=5);
}
void write()
{
             int data,i;
             FILE *ptr;
             ptr=fopen("stack.txt","w");
             for(i=top;i>=0;i--)
                          fprintf(ptr,"%d\t",stack[i]);
             fclose(ptr);
}
void read()
{
             int i;
             FILE *fptr;
             fptr=fopen("stack.txt","r");
            for(i=top;i>=0;i--)
                        fscanf(fptr,"%d\t",&stack[i]);
            fclose(fptr);


}
void push()
{
    int data;
    if(top>=size-1)
    {
        printf("\n stack is overflow");
    }
    else
    {
            printf("\Enter an element=");
                        scanf("%d",&data);
        top++;
        stack[top]=data;

    }
}



void peep()
{
    int i;
    if(top==-1)
    {
        printf("\n stack is underflow");
    }
    else
    {
        for(i=top;i>=0;i--)
        {
            printf("\n %d",stack[i]);

        }
    }
}

void pop()
{
    if(top==-1)
    {
        printf("\n stack is underflow");

    }
    else
    {
        printf("\n popped element is=%d",stack[top]);
        stack[top--];
    }
}


void update()
{
    int u,n;
    if(top==-1)
    {
        printf("\n stack is underflow");
    }
    else
    {
        printf("\n enter position  at you want to change=");
        scanf("%d",&u);
        printf("\n enter new element= ");
        scanf("%d",&n);
        stack[u]=n;

    }
}

void isEmpty()
{
    if(top==-1)
    {
        printf("\n stack is empty");
    }
    else
    {
        printf("\n stack is not empty");
    }
}
```

Output:



```
 enter your choice=1
�nter an element=25

 enter your choice=1
�nter an element=35

 enter your choice=1
�nter an element=65

 enter your choice=1
�nter an element=95

 enter your choice=2

 popped element is=95
 enter your choice=2

 popped element is=65
 enter your choice=3

 35
 25
 enter your choice=4

 enter position  at you want to change=1

 enter new element= 65

 enter your choice=5

 stack is not empty
Process returned 0 (0x0)   execution time : 46.378 s
Press any key to continue.
```

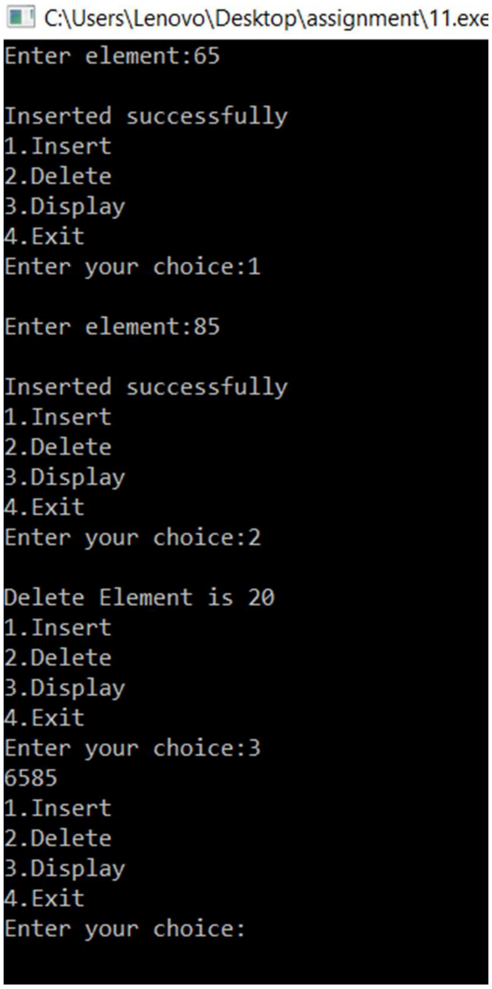
11. Write a program to implement Queue with required operations using array.

```
#include<stdio.h>
#include<conio.h>
#include<ctype.h>
#include<string.h>
#include<stdlib.h>
#define size 10
int ch;
int q[size];
int rear=-1;
int front=-1;
void insert_queue();
void delete_queue();
void disp_queue();
void insert_queue()
{
        if(rear>=size)
                printf("\nQueue is overflow");
        else
        {
                if(front==-1)
                        front=0;
                printf("\nEnter element:");
        scanf("%d",&ch);
                rear=rear+1;
                q[rear]=ch;
                printf("\nInserted successfully");
        }

}
void delete_queue()
{
        if(front==-1 || front>=rear)
                printf("\nQueue underflow");
        else
        {
                ch=q[front];
                printf("\nDelete Element is %d",ch);
                front=front+1;
        }
}
void disp_queue()
{
        int i;
        if(front==-1)
                return;
        for(i=front;i<=rear;i++)
                printf("%d",q[i]);
}
void write()
{
        int data,i;
        FILE *ptr;
        ptr=fopen("queue.txt","w");
        for(i=front;i<=rear;i++)
                fprintf(ptr,"%d\t",q[i]);
        fclose(ptr);
}
void read()
{
        int i;
        FILE *fptr;
        fptr=fopen("queue.txt","r");
        for(i=front;i<=rear;i++)
                fscanf(fptr,"%d\t",&q[i]);
        fclose(fptr);
        printf("\nRead successufully");
```

```
}
void main()
{
    read();
    int ch;
    do
    {
        printf("\n1.Insert");
        printf("\n2.Delete");
        printf("\n3.Display");
        printf("\n4.Exit");
        printf("\nEnter your choice:");
        scanf("%d",&ch);
        switch(ch)
        {
            case 1:insert_queue();
                    write();
                    break;
            case 2:delete_queue();
                    write();
                    break;
            case 3:disp_queue();
                    break;
            case 4:exit(1);
                    break;
            default:
                    break;
        }
    }while(ch!=5);
```

```
}
```

Output:



C:\Users\Lenovo\Desktop\assignment\11.exe

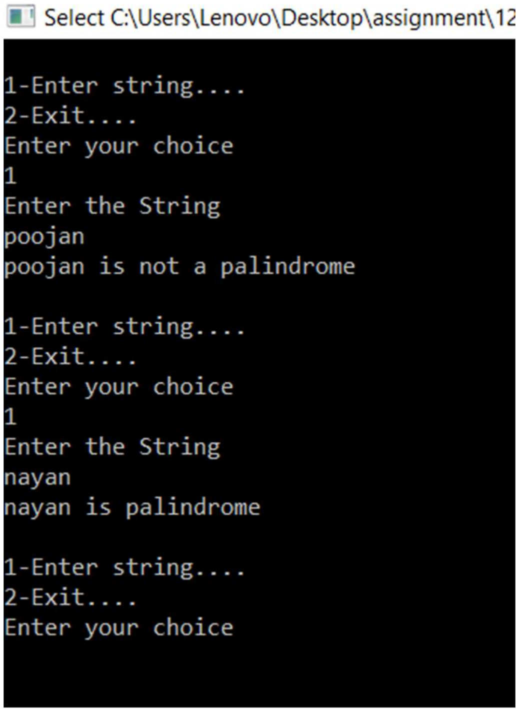
```
Enter element:65

Inserted successfully
1.Insert
2.Delete
3.Display
4.Exit
Enter your choice:1

Enter element:85

Inserted successfully
1.Insert
2.Delete
3.Display
4.Exit
Enter your choice:2

Delete Element is 20
1.Insert
2.Delete
3.Display
4.Exit
Enter your choice:3
6585
1.Insert
2.Delete
3.Display
4.Exit
Enter your choice:
```

12. Write a program to check whether the string is palindrome or not. Use Stack Data Structure for the same.

```
#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#define MAX 50


int top = -1, front = 0;
int stack[MAX];
void push(char);
void pop();

void main()
{
//          read();
            FILE *ptr;
   int i, ch;
   char s[MAX], b;
   do
   {
      printf("\n1-Enter string....");
      printf("\n2-Exit....");
      printf("\nEnter your choice\n");
      scanf("%d",&ch);
      switch (ch)
      {
          case 1:
                      ptr=fopen("palindrome.txt","w");
          printf("Enter the String\n");
          scanf("%s", s);
          for (i = 0;s[i] != '\0';i++)
          {
                      b = s[i];
                      push(b);
          }
          for (i = 0;i < (strlen(s) / 2);i++)
          {
                      if (stack[top] == stack[front])
                      {
                      pop();
                      front++;
                      }
                      else
                      {
                      printf("%s is not a palindrome\n", s);
                      fprintf(ptr,"%s is not palindrome",s);
                      break;
                      }
          }

          if ((strlen(s) / 2)  ==  front)
          {
                      printf("%s is palindrome\n",  s);
                      fprintf(ptr,"%s is palindrome",s);
          }
          front  =  0;
          top  =  -1;
          fclose(ptr);
          //write();
          break;
          case 2:
          exit(0);
          default:
          printf("enter correct choice\n");
      }
   }while(ch!=3);
```

```
}


void push(char a)
{
   top++;
   stack[top]  =  a;
}
void pop()
{
   top--;
}
```

Output:



Select C:\Users\Lenovo\Desktop\assignment\12

```

1-Enter string....
2-Exit....
Enter your choice
1
Enter the String
poojan
poojan is not a palindrome

1-Enter string....
2-Exit....
Enter your choice
1
Enter the String
nayan
nayan is palindrome

1-Enter string....
2-Exit....
Enter your choice
```

13. Write a program to implement Doubly Linked List.

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

```
#include<conio.h>

#include<stdlib.h>

struct node

{

	int data;

	struct node *next;

	struct node *prev;

};

struct node *head=NULL;


void insert_first()

{

	int n;

	struct node *new_node;

	new_node=(struct node*)malloc(sizeof(struct node));

	printf("\nEnter number:");

	scanf("%d",&n);

	new_node->data=n;

	new_node->next=NULL;

	new_node->prev=NULL;

	if(head!=NULL)

	{

		new_node->next=head;

		head->prev=new_node;

		head=new_node;

	}

	else

		head=new_node;

}
```

```
void insert_last()
{
	int n;
	struct node *new_node;
	struct node *temp;
	new_node=(struct node*)malloc(sizeof(struct node));
	printf("\nEnter number:");
	scanf("%d",&n);
	new_node->data=n;
	new_node->next=NULL;
	new_node->prev=NULL;
	temp=head;
	if(head==NULL)
		printf("\nList is emtpty");
	else
	{
		while(temp->next!=NULL)
			temp=temp->next;
		temp->next=new_node;
		new_node->prev=temp;
	}
}
void insert_specific()
{
	int n;
	struct node *new_node;
	struct node *temp;
	int p,cnt=1;
	new_node=(struct node*)malloc(sizeof(struct node));
```

```
    printf("\nEnter number:");
    scanf("%d",&n);
    new_node->data=n;
    new_node->next=NULL;
    new_node->prev=NULL;
    printf("\nEnter position:");
    scanf("%d",&p);
    temp=head;
    if(head==NULL)
    {
        printf("\nList is empty");
    }
    else
    {
        while(cnt!=p)
        {
            temp=temp->next;

            cnt++;
        }
        new_node->next=temp->next;
        temp->next=new_node;
        new_node->prev=temp;
    }
}
void insert()
{
    int ch,inch;
    printf("\n1.Insert First....");
    printf("\n2.Insert Last.....");
    printf("\n3.Insert Specific..");
    printf("\nEnter your choice...");
    scanf("%d",&ch);
    if(ch==1)
        insert_first();
    else if(ch==2)
        insert_last();
    else
        insert_specific();
}
void delete_first()
{
    struct node *temp;
    if(head==NULL)
        printf("\nList is empty");
    else
    {
        temp=head;
        head=head->next;
        head->next->prev=head;
        printf("\nDeleted element is %d",temp->data);
        free(temp);
    }

}
void delete_last()
{
    struct node *temp;
```

```
        struct node *prev;
        temp=head;
        if(head==NULL)
                printf("\nList is empty");
        else
        {
                while(temp->next!=NULL)
                {
                        prev=temp;
                        temp=temp->next;
                }
                printf("\nDeleted element is %d",temp->data);
                prev->next=NULL;
                free(temp);
        }
}
void delete()
{
        int ch,inch;
        printf("\n1.Delete First....");
        printf("\n2.Delete Last.....");
        printf("\n3.Delete Specific..");
        printf("\nEnter your choice...");
        scanf("%d",&ch);
        if(ch==1)
                delete_first();
        else if(ch==2)
                delete_last();
        //else
        //      delete_specific();
}
void disp()
{
        struct node *new_node;
        struct node *temp;
        temp=head;
        while(temp)
        {
                printf("\ndata is %d",temp->data);
                temp=temp->next;
        }
}
void write()
{
        int data,i;
        FILE *ptr;
        ptr=fopen("doublylink.txt","w");
        struct node *temp;
        temp=head;
        while(temp)
        {
                fprintf(ptr,"%d",temp->data);
                temp=temp->next;
        }
        fclose(ptr);
}
void read()
{
        int i;
        FILE *fptr;
        fptr=fopen("doublylink.txt","r");
        struct node *temp;
        temp=head;
        while(temp)
        {
                fscanf(fptr,"%d",&temp->data);
                temp=temp->next;
        }
        fclose(fptr);
        printf("\nRead successufully");
}
int main()
{
        read();
        int ch,n;
        do
        {
                printf("\n1.Insert..");
                printf("\n2.Delete..");
                printf("\n3:Display");
                printf("\n4:Exit");
                printf("\nEnter your choice:");
                scanf("%d",&ch);
                switch(ch)
                {
                    case 1:insert();
                            write();
```

```
                                        break;
                        case 2:delete();
                                        write();
                                        break;
                        case 3:disp();
                                        break;
                        case 4:exit(1);
                                  break;
                        default:
                                        break;
                }
        }while(ch!=4);
        return 0;
}
```

Output:

```
Read successufully
1.Insert..
2.Delete..
3:Display
4:Exit
Enter your choice:1

1.Insert First....
2.Insert Last.....
3.Insert Specific..
Enter your choice...1
Enter number:20

1.Insert..
2.Delete..
3:Display
4:Exit
Enter your choice:1

1.Insert First....
2.Insert Last.....
3.Insert Specific..
Enter your choice...2

Enter number:65

1.Insert..
2.Delete..
3:Display
4:Exit
Enter your choice:3

data is 20
data is 65
1.Insert..
2.Delete..
3:Display
4:Exit
Enter your choice:2
```

```
1.Delete First....
2.Delete Last.....
3.Delete Specific..
Enter your choice...2

Deleted element is 65
1.Insert..
2.Delete..
3:Display
4:Exit
Enter your choice:3

data is 20
1.Insert..
2.Delete..
3:Display
4:Exit
Enter your choice:
```