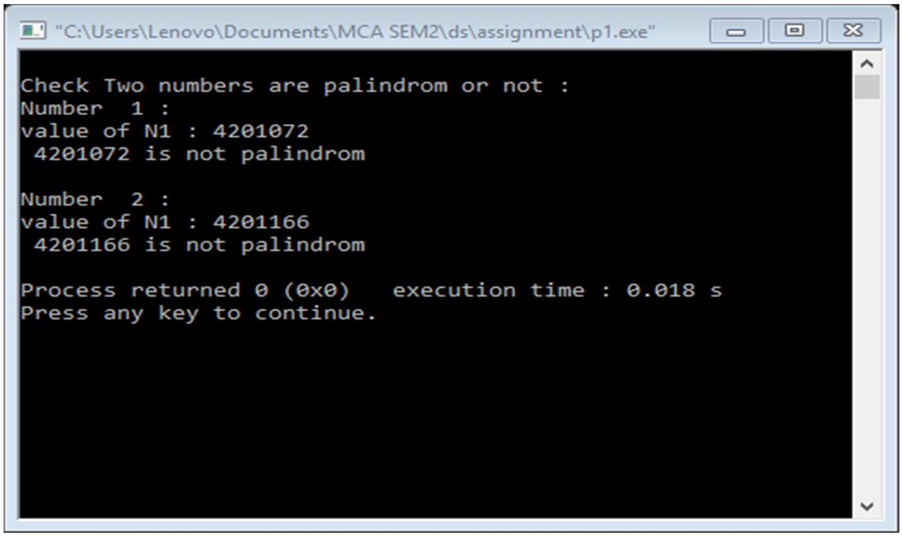
ASSIGNMENT

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.

- PROGRAM

```
#include<stdio.h>
#include<string.h>
void swap(int *x1,int *x2)
{
   int temp;
   temp=*x1;
   *x1=*x2;
   *x2=temp;
}
void palindrom(int x1)
{
   int n1=x1;
   int d=0;
   printf("value of N1 : %d\n",x1);
   while(x1 != 0)
   {
      d = d * 10;
      d = d + x1%10;
      x1= x1/10;
   }
   if(n1 == d)
   {
      printf(" %d is Palindrom \n",n1);
   }
   else
   {
      printf(" %d is not palindrom \n",n1);
   }
}
void read_from_file()
{
   FILE *in_file;
   int n,no[2];

   in_file = fopen("swap.txt","r");
   if(in_file == NULL)
   {
      printf("error\n");
    }

    int i=0;
    while(fscanf(in_file,"%d",&n) != EOF)
    {
      no[i]=n;
      i++;
    }
    fclose(in_file);
}

int main()
{
    int n1,n2,no[2];
    read_from_file();
    n1=no[0];
    n2=no[1];
    void (*p)(int,int)=&swap;
    (*p)(&n1,&n2);
    printf("\n");
    printf("Check Two numbers are palindrom or not :\n");
    printf("Number  1 : \n");
    palindrom(n1);
    printf("\n");
    printf("Number  2 : \n");
    palindrom(n2);
}
```

- OUTPUT



```
Check Two numbers are palindrom or not :
Number  1 :
value of N1 : 4201072
 4201072 is not palindrom

Number  2 :
value of N1 : 4201166
 4201166 is not palindrom

Process returned 0 (0x0)   execution time : 0.018 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}}.

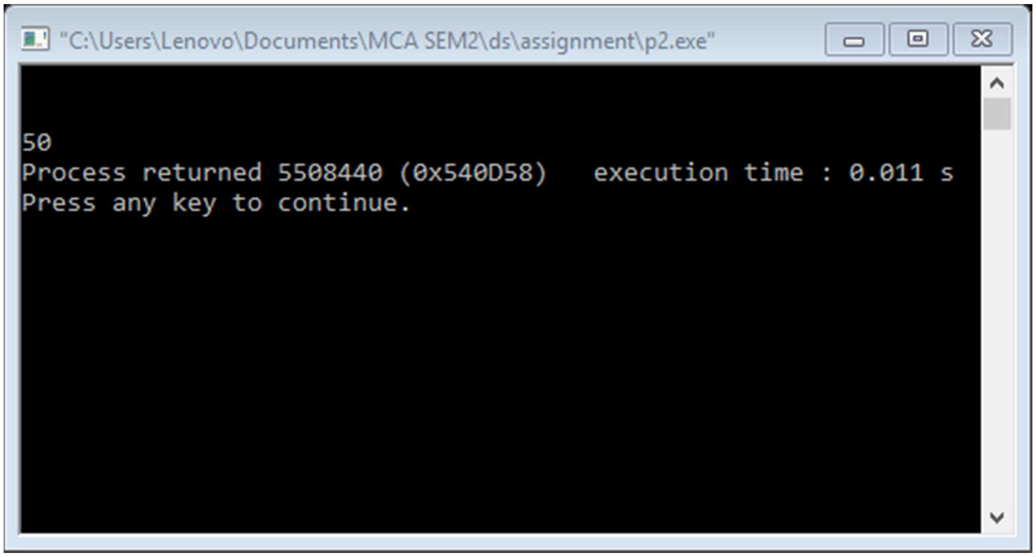
- PROGRAM

```
#include<stdio.h>
#include<stdlib.h>
struct node{
   int data;
   struct node *next;
}*head=NULL;
int c=0;
void insert(int data)
{
   struct node *temp,*newnode;
   newnode=(struct node*)malloc(sizeof(struct node));
   newnode->data=data;
   if(head == NULL)
   {
      head=newnode;
      newnode->next=NULL;
   }
   else
   {
      temp=head;
      while(temp->next != NULL)
      {
         temp=temp->next;
      }
      temp->next=newnode;
      newnode->next=NULL;
   }
}

void display()
{
   struct node *temp;
   if(head == NULL)
   {
      printf("LINK LIST IS EMPTY\n");
   }
   temp=head;
   while(temp != NULL)
```

```
    {
        printf("%d ->",temp->data);
        temp=temp->next;
    }
}
int length()
{
    struct node *temp;
    temp=head;
    while(temp != NULL)
    {
        c++;
        temp=temp->next;
    }
    return c;
}

void powerset(struct node *v, struct node *up)
{
    if(v != NULL)
    {
            printf("%d",head->data);
            head=head->next;
    }
    else
    {
        struct node *t,*q;
        t=v->next;
        q=up->next;
        powerset(t,q);
        powerset(t,q);
    }
}

void main()
{
    insert(50);
    insert(60);
    printf("\n\n");
    struct node *list;
    list=head;
    int n;
    n=length();
    powerset(list,head);
```

```
}
```

- OUTPUT



```

50
Process returned 5508440 (0x540D58)   execution time : 0.011 s
Press any key to continue.
```

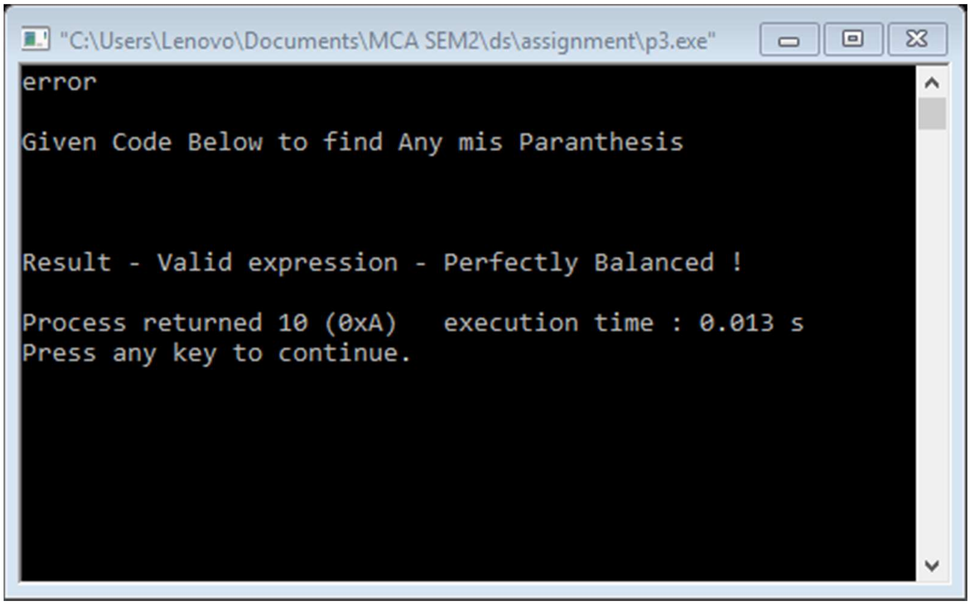
3. Write a program to check the balance of parenthesis if an expression. Implement required data structure for the same.

- PROGRAM

```
#include <stdio.h>
#include <string.h>
#define MAX_SIZE 100

int top=-1;
char arr[MAX_SIZE];

int isEmpty(){
   if(top == -1){
      return 1;
   }else{
      return 0;
   }
}

int isFull(){
   if(top == MAX_SIZE-1){
      return 1;
   }else{
      return 0;
   }
}

void push(char item){
    if(isFull())
    {
            printf("Stack is full");
    }
    else
    {
        top++;
        arr[top] = item;
    }
}

void pop(){
    if(isEmpty()){
        printf("Stack is empty");
    }else{
        top--;
    }
}

char gettop()
{
    return arr[top];
}

int ArePair(char opening,char closing)
{
            if(opening == '(' && closing == ')') return 1;
            else if(opening == '{' && closing == '}') return 1;
            else if(opening == '[' && closing == ']') return 1;
            return 0;
}


void read_from_file()
{
    FILE *in_file;
    char in_expr;

    in_file = fopen("parenthesis.txt","r");

    if(in_file == NULL)
    {
        printf("error\n");
    }
```

```
  printf("\nGiven Code Below to find Any mis Paranthesis \n\n");
  while(fscanf(in_file,"%c",&in_expr) != EOF)
  {
    printf("%c",in_expr);
             if(in_expr == '(' || in_expr == '{' || in_expr == '[')
             {
        push(in_expr);
    }
             else if(in_expr == ')' || in_expr == '}' || in_expr == ']')
             {
      char a = gettop();
      if(isEmpty() || !ArePair(gettop(),in_expr))
                    {
        printf("\nResult - Invalid expression - Not a Balanced one !");
        return 0;
      }
                    else
                    {
        pop();
      }
    }
  }
  fclose(in_file);
}
void main()
{

  read_from_file();

  if(isEmpty()){
    printf("\n\nResult - Valid expression - Perfectly Balanced !");
  }else{
    printf("\n\nResult - Invalid expression - Not a Balanced one !");
  }
  printf("\n");
}
```

- OUTPUT



```
error

Given Code Below to find Any mis Paranthesis



Result - Valid expression - Perfectly Balanced !

Process returned 10 (0xA)   execution time : 0.013 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.

- PROGRAM

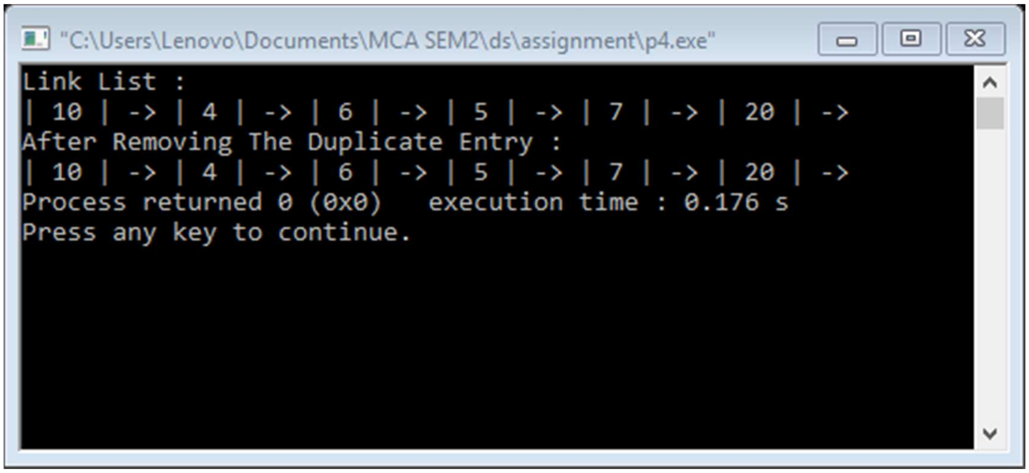
```
#include<stdio.h>
#include<stdlib.h>
struct node{
   int data;
   struct node*next;
}*head=NULL;
void insert(int n)
{
   struct node *temp,*newnode;
   newnode=(struct node*)malloc(sizeof(struct node));
   newnode->data=n;
   if(head == NULL)
   {
      head=newnode;
      newnode->next=NULL;
   }
   else
   {
      temp=head;
      while(temp->next != NULL)
         temp=temp->next;
      temp->next=newnode;
      newnode->next=NULL;
   }
```

```
}

void read_from_file()
{
   FILE *in_file;
   int n;

   in_file = fopen("Link_list.txt","r");

   if(in_file == NULL)
   {
      printf("error\n");
   }
   while(fscanf(in_file,"%d",&n) != EOF)
   {
     insert(n);
   }
   fclose(in_file);
}

void write_into_file()
{
          FILE *out_file;

   out_file = fopen("Link_list.txt","w");

   if(out_file == NULL)
   {
      printf("error\n");
   }
   struct node * temp;
   temp=head;
   while(temp != NULL)
   {
      fprintf(out_file,"%d\n",temp->data);
      temp=temp->next;
   }
   fclose(out_file);
}

void find_duplicate()
{
   struct node *temp ,*temp1 ,*dup;
   temp=head;
```

```
    while(temp != NULL)
    {
         temp1=temp;
         while(temp1->next != NULL)
         {
             if(temp->data == temp1->next->data)
             {
                 dup=temp1->next;
                 temp1->next=temp1->next->next;
                 free(dup);
             }
             else
             {
                 temp1=temp1->next;
             }
         }
         temp=temp->next;
    }
}

void display()
{
    struct node *temp;
    if(head == NULL)
    {
        printf("List is Empty \n");
        return;
    }
    temp=head;
    while(temp != NULL)
    {
         printf("| %d | -> ",temp->data);
         temp=temp->next;
    }
}
void main()
{
   read_from_file();
   printf("Link List : \n");
   display();
   find_duplicate();
   write_into_file();
   printf("\nAfter Removing The Duplicate Entry :\n");
   display();
```

}

- OUTPUT



```
Link List :
| 10 | -> | 4 | -> | 6 | -> | 5 | -> | 7 | -> | 20 | ->
After Removing The Duplicate Entry :
| 10 | -> | 4 | -> | 6 | -> | 5 | -> | 7 | -> | 20 | ->
Process returned 0 (0x0)   execution time : 0.176 s
Press any key to continue.
```

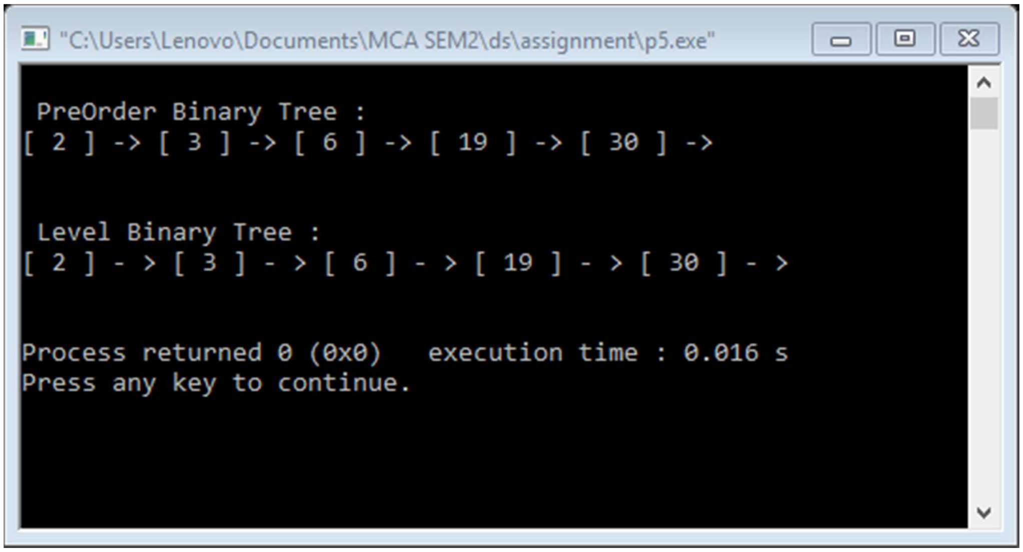
5. Write a program to create a binary tree. Implement required method to generate a binary tree from user inputs and to display binary tree using level order and pre order traversals

➢ PROGRAM

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


struct node* root;

struct node* insert(struct node* r, int data)
{
   if(r==NULL)
   {
      r = (struct node*) malloc(sizeof(struct node));
      r->data = data;
      r->left = NULL;
      r->right = NULL;
      return r;
   }
   else if(data < r->data){
      r->left = insert(r->left, data);
   }
   else {
      r->right = insert(r->right, data);
   }
   return r;

}

void Preorder(struct node *root)
{
   if(root != NULL)
   {
      printf("[ %d ] -> ",root->data);
      Preorder(root->left);
      Preorder(root->right);
   }
}

//read data from file
void read_from_file()
{
   FILE *in_file;
   int n;

   in_file = fopen("BSTtree.txt","r");

   if(in_file == NULL)
   {
      printf("error101\n");
   }
   while(fscanf(in_file,"%d",&n) != EOF)
   {
                  root=insert(root,n);
   }
   fclose(in_file);
}

int queue[100];
int front=0;
int rear=-1;

void enQueue(struct node *new_node)
{
          queue[rear++] = new_node;
}
```

```
struct node *deQueue()
{
          if(front == rear)
   {
      return NULL;
   }
   else
   {
      return queue[front++];
   }
}

void printLevelOrder(struct node* root)
{
          struct node *temp_node = root;
          enQueue(root);
          while (temp_node != NULL)
          {
                    printf("[ %d ] - > ", temp_node->data);

                    if (temp_node->left != NULL)
      {
         enQueue(temp_node->left);
      }

                    if (temp_node->right != NULL)
      {
         enQueue(temp_node->right);
      }
                    temp_node = deQueue();
          }
}

int main()
{
          read_from_file();
          printf("\n PreOrder Binary Tree : \n");
          Preorder(root);
          printf("\n\n");
          printf("\n Level Binary Tree : \n");
          printLevelOrder(root);
          printf("\n\n");
}
```

- OUTPUT



```
 PreOrder Binary Tree :
[ 2 ] -> [ 3 ] -> [ 6 ] -> [ 19 ] -> [ 30 ] ->


 Level Binary Tree :
[ 2 ] - > [ 3 ] - > [ 6 ] - > [ 19 ] - > [ 30 ] - >


Process returned 0 (0x0)   execution time : 0.016 s
Press any key to continue.
```

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).

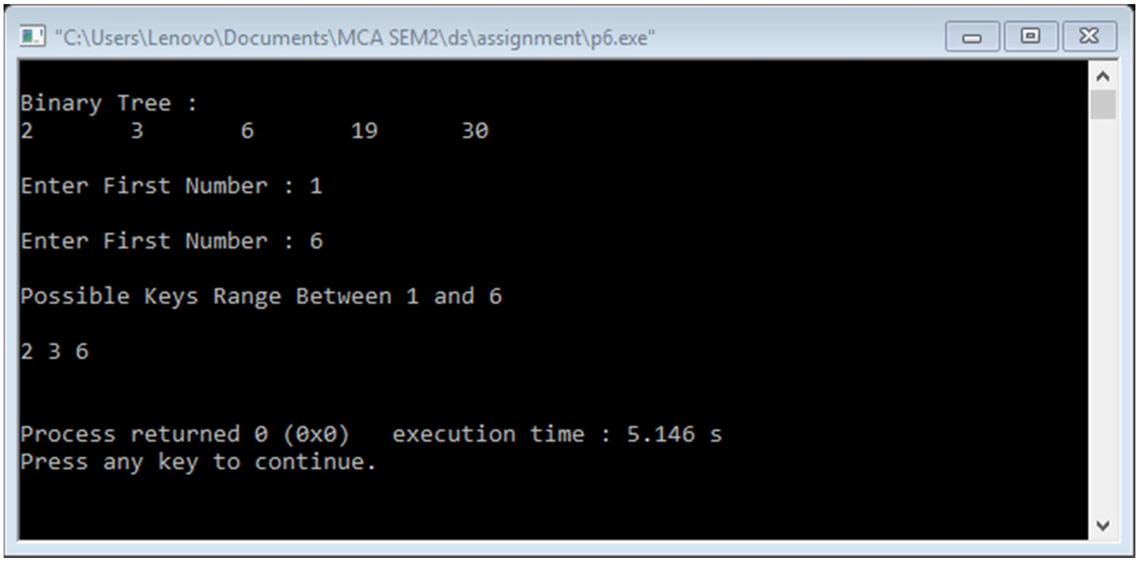
➢ PROGRAM

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


struct node* root;

struct node* insert(struct node* r, int data)
{
  if(r==NULL)
  {
    r = (struct node*) malloc(sizeof(struct node));
    r->data = data;
    r->left = NULL;
    r->right = NULL;
    return r;
  }
```

```
   else if(data < r->data){
      r->left = insert(r->left, data);
   }
   else {
      r->right = insert(r->right, data);
   }
   return r;

}

void Print(struct node *root, int k1, int k2)
{

          if ( NULL == root )
                    return;

          if ( k1 < root->data )
                    Print(root->left, k1, k2);

          if ( k1 <= root->data && k2 >= root->data )
                    printf("%d ", root->data );

          if ( k2 > root->data )
                    Print(root->right, k1, k2);
}

//read data from file
void read_from_file()
{
   FILE *in_file;
   int n;

   in_file = fopen("BSTtree.txt","r");

   if(in_file == NULL)
   {
      printf("error101\n");
   }
   while(fscanf(in_file,"%d",&n) != EOF)
   {
                    root=insert(root,n);
   }
   fclose(in_file);
}
```

```
void Display(struct node* root)
{
   if(root != NULL)
   {
      printf("%d \t",root->data);
      Display(root->left);
      Display(root->right);
   }
}

int main()
{
   int k1,k2;
          read_from_file();
          printf("\nBinary Tree : \n");
          Display(root);
          printf("\n\nEnter First Number : ");
          scanf("%d",&k1);
          printf("\nEnter First Number : ");
          scanf("%d",&k2);
          printf("\n");
          printf("Possible Keys Range Between %d and %d \n",k1,k2);
          printf("\n");
          Print(root,k1,k2);
          printf("\n\n");
}
```

- OUTPUT



```

Binary Tree :
2       3       6       19      30

Enter First Number : 1

Enter First Number : 6

Possible Keys Range Between 1 and 6

2 3 6


Process returned 0 (0x0)   execution time : 5.146 s
Press any key to continue.
```

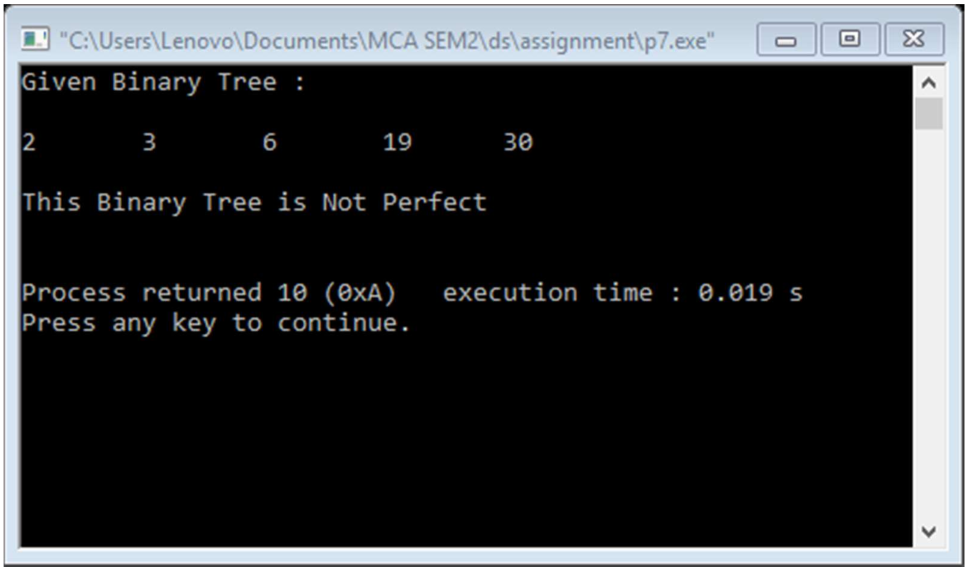
7. Write a program to create a binary tree. Implement required method to generate a binary tree from user inputs and check whether the Binary Tree is a perfect binary tree.

- PROGRAM

```
#include<stdio.h>
#include<stdlib.h>
struct tnode{
   int data;
   struct tnode *lchiled;
   struct tnode *rchiled;
};

//insert new node
struct tnode *Create(struct tnode *p,int value)
{

   struct tnode *temp,*temp1;
   //For create Root node
   if(p == NULL)
   {
      p=(struct tnode*)malloc(sizeof(struct tnode));
      if(p == NULL)
      {
         printf("Error : Allocating Memory \n");
         exit(0);
      }
      else
      {
         p->data = value;
         p->rchiled=NULL;
         p->lchiled=NULL;
      }
   }
   //if Root Node Exit
   else
   {
      temp=p;

      while(temp != NULL)
      {
         temp1=temp;
         if(temp1->data > value)
         {
            temp=temp->lchiled;
         }
         else
         {
            temp=temp->rchiled;
         }
      }

      if(temp1->data > value)
      {
         temp1->lchiled=(struct tnode*)malloc(sizeof(struct tnode));
         temp1=temp1->lchiled;
         if(temp1 == NULL)
         {
            printf("Error : Allocating Memory \n");
            exit(0);
         }
         else
         {
            temp1->data=value;
            temp1->rchiled=temp1->lchiled=NULL;
         }
      }
      else
      {
         temp1->rchiled=(struct tnode*)malloc(sizeof(struct tnode));
         temp1=temp1->rchiled;
         if(temp1 == NULL)
         {
            printf("Error : Allocating Memory \n");
            exit(0);
         }
         else
         {
            temp1->data=value;
            temp1->rchiled=temp1->lchiled=NULL;
         }
      }
   }
   return(p);
}

int findADepth(struct tnode *node)
{
  int d = 0;
  while (node != NULL)
  {
    d++;
    node = node->lchiled;
  }
  return d;
}
int isPerfectRec(struct tnode* root, int d, int level)
{

   if (root == NULL)
      return 1;


   if (root->lchiled == NULL && root->rchiled == NULL)
      return (d == level+1);

   if (root->lchiled == NULL || root->rchiled == NULL)
      return 0;

   return isPerfectRec(root->lchiled, d, level+1) &&
        isPerfectRec(root->rchiled, d, level+1);
}

int isPerfect(struct tnode *root)
{
  int d = findADepth(root);
  return isPerfectRec(root,d,0);
}
struct tnode *root;
//read data from file
void read_from_file()
{
   FILE *in_file;
   int n;

   in_file = fopen("BSTtree.txt","r");

   if(in_file == NULL)
   {
      printf("error101\n");
   }
   printf("Given Binary Tree : \n\n");
```

```
    while(fscanf(in_file,"%d",&n) != EOF)
    {
        printf("%d\t",n);
                    root=Create(root,n);
    }
    fclose(in_file);
}

void  main()
{
    read_from_file();
    printf("\n\n");
    if (isPerfect(root))
        printf("This Binary Tree is Perfect \n");
    else
        printf("This Binary Tree is Not Perfect \n");
    printf("\n");
}
```

- OUTPUT



```
Given Binary Tree :

2       3       6       19      30

This Binary Tree is Not Perfect


Process returned 10 (0xA)   execution time : 0.019 s
Press any key to continue.
```

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

➢ PROGRAM

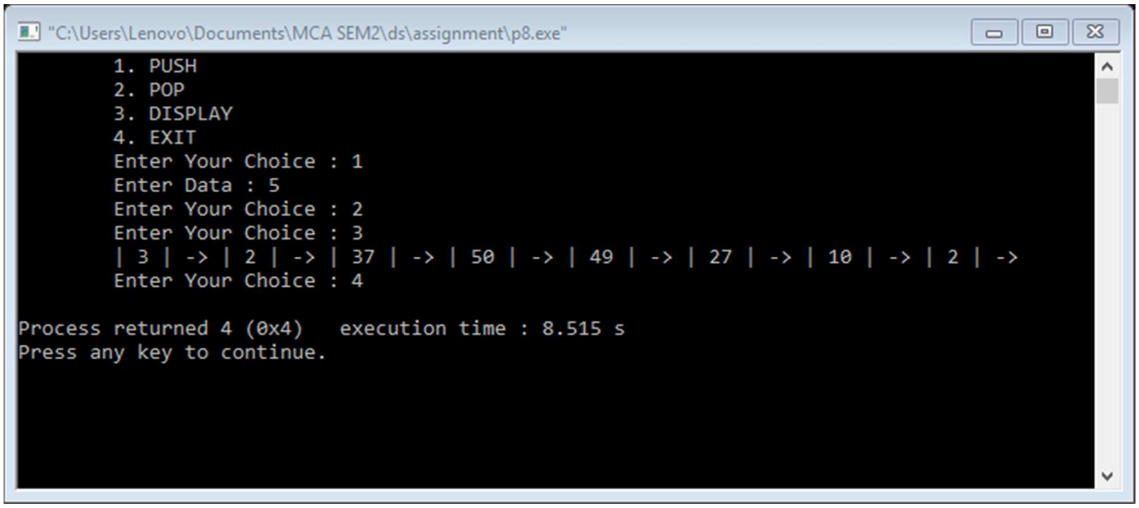
```
#include<stdio.h>
#include<stdlib.h>
struct node{
    int data;
    struct node *next;
}*head=NULL;
```

```
}

void push(int item)
{
   struct node *newnode;
   newnode=(struct node*)malloc(sizeof(struct node));
   newnode->data=item;
   if(head == NULL)
   {
      head=newnode;
      newnode->next=NULL;
   }
   else
   {
      newnode->next=head;
      head=newnode;
   }
}
void pop()
{
   struct node *temp;
   if(head==NULL)
   {
      printf("LIST IS EMPTY\n");
   }
   temp=head;
   head=temp->next;
   free(temp);
}

void Display()
{
   struct node *temp;
   temp=head;
   if(head == NULL)
   {
      printf("LINK LIST IS EMPTY\n");
   }
   printf("\t");
   while(temp != NULL)
   {
      printf("| %d | -> ",temp->data);
      temp=temp->next;
   }
   printf("\n");
}
```

```
void write_into_file()
{
          FILE *out_file;

   out_file = fopen("number.txt","w");

   struct node *temp;
   temp=head;

   if(head==NULL)
   {
      printf("\tQueue is Empty \n");
   }

   while(temp != NULL)
   {
      fprintf(out_file,"%d\n",temp->data);
      temp=temp->next;
   }
   fclose(out_file);
}

void read_from_file()
{
   FILE *in_file;
   int n;

   in_file = fopen("number.txt","r");
   if(in_file == NULL)
   {
      printf("error\n");
   }
   while(fscanf(in_file,"%d",&n) != EOF)
   {
     // fscanf(in_file,"%d",&n1);
     push(n);
   }
   fclose(in_file);
}

void main()
{
   read_from_file();
```

```
    int ch,item;
    printf("\t1. PUSH\n");
    printf("\t2. POP\n");
    printf("\t3. DISPLAY\n");
    printf("\t4. EXIT\n");
    do{
        printf("\tEnter Your Choice : ");
        scanf("%d",&ch);
        switch(ch)
        {
            case 1:
                printf("\tEnter Data : ");
                scanf("%d",&item);
                push(item);
                break;
            case 2:
                pop();
                write_into_file();
                break;
            case 3:
                 Display();
                 break;
            case 4:
                 break;
        }
    }while(ch!=4);
}
```

- OUTPUT



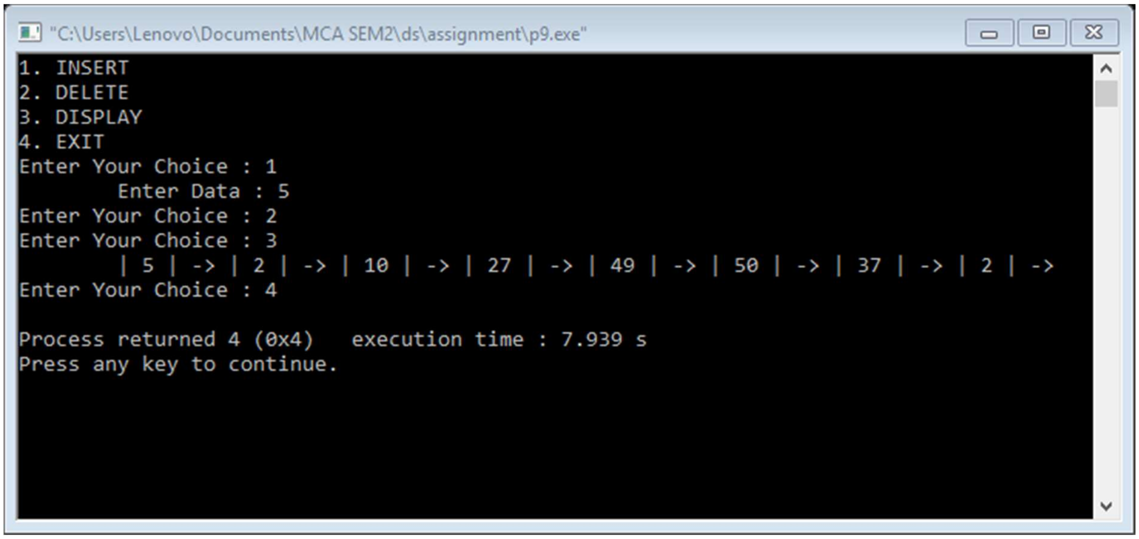
```
        1. PUSH
        2. POP
        3. DISPLAY
        4. EXIT
        Enter Your Choice : 1
        Enter Data : 5
        Enter Your Choice : 2
        Enter Your Choice : 3
        | 3 | -> | 2 | -> | 37 | -> | 50 | -> | 49 | -> | 27 | -> | 10 | -> | 2 | ->
        Enter Your Choice : 4

Process returned 4 (0x4)   execution time : 8.515 s
Press any key to continue.
```

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

➢ PROGRAM

```
#include<stdio.h>
struct node{
   int data;
   struct node *next;
}*head=NULL;
void insert(int item)
{
   struct node *newnode;
   newnode=(struct node*)malloc(sizeof(struct node));
   newnode->data=item;
   if(head == NULL)
   {
      head=newnode;
      newnode->next=NULL;
   }
   else
   {
      newnode->next=head;
      head=newnode;
   }
}
void Delete()
{
   struct node *temp;
   if(head == NULL)
   {
      printf("Queue is Empty \n");
   }
   temp=head;
   while(temp->next->next != NULL)
   {
      temp=temp->next;
   }
   temp->next=NULL;
}
void display()
{
   struct node *temp;
   temp=head;
   if(head==NULL)
   {
      printf("Queue is Empty \n");
   }
   printf("\t");
   while(temp != NULL)
   {
      printf("| %d | -> ",temp->data);
      temp=temp->next;
   }
   printf("\n");
}

void write_into_file()
{
   //printf("write_into_file()\n");
          FILE *out_file;

   out_file = fopen("number.txt","w");

   struct node *temp;
   temp=head;

   if(head==NULL)
   {
      printf("\tQueue is Empty \n");
   }

   while(temp != NULL)
   {
      fprintf(out_file,"%d\n",temp->data);
      temp=temp->next;
   }
   fclose(out_file);
}

void read_from_file()
{
   FILE *in_file;
   int n;

   in_file = fopen("number.txt","r");
   if(in_file == NULL)
   {
      printf("error\n");
   }
   while(fscanf(in_file,"%d",&n) != EOF)
   {
     // fscanf(in_file,"%d",&n1);
     insert(n);
   }
   fclose(in_file);
}

void main()
{
   read_from_file();
   int ch,item;
   printf("1. INSERT\n");
   printf("2. DELETE\n");
   printf("3. DISPLAY\n");
   printf("4. EXIT\n");
   do{
      printf("Enter Your Choice : ");
      scanf("%d",&ch);
      switch(ch)
      {
         case 1:
            printf("\tEnter Data : ");
            scanf("%d",&item);
            insert(item);
            break;
         case 2:
            Delete();
            write_into_file();
            break;
         case 3:
             display();
             break;
         case 4:
             break;
      }
   }while(ch!=4);
}
```

- OUTPUT



```
1. INSERT
2. DELETE
3. DISPLAY
4. EXIT
Enter Your Choice : 1
        Enter Data : 5
Enter Your Choice : 2
Enter Your Choice : 3
        | 5 | -> | 2 | -> | 10 | -> | 27 | -> | 49 | -> | 50 | -> | 37 | -> | 2 | ->
Enter Your Choice : 4

Process returned 4 (0x4)   execution time : 7.939 s
Press any key to continue.
```

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

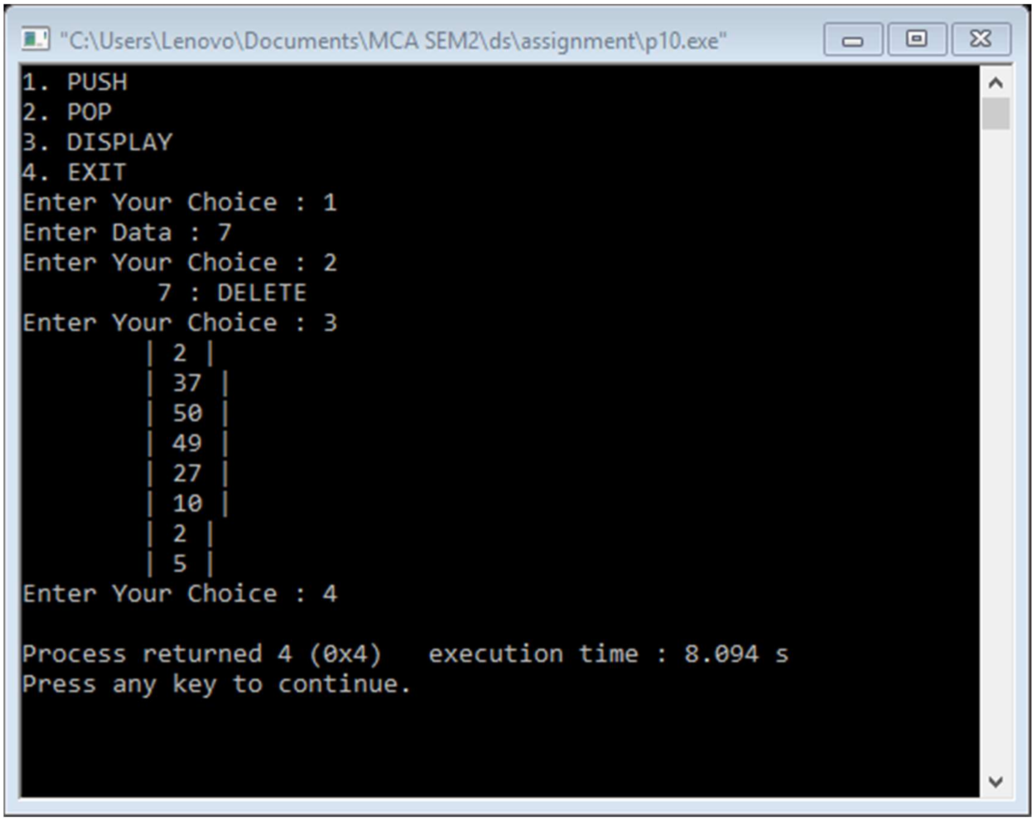
➢ PROGRAM

```
#include<stdio.h>
#define SIZE 100
int stack[SIZE];
int top=-1;

void push(int item)
{
        if(top >= SIZE-1)
        {
                printf("\nStack Overflow.");
        }
        else
        {
                top = top+1;
                stack[top] = item;
                write_into_file();
        }
}

int pop()
{
   int item;
        if(top <0)
        {
                printf("stack under flow:");
   }
```

```
          else
          {
                    item = stack[top];
                    printf("\t %d : DELETE\n",item);
                    top = top-1;
                    return(item);
          }
}
void display()
{
   if(top==-1)
   {
      printf("\tSTACK IS EMPTY\n");
   }
   int i;
   for(i=top;i>=0;i--)
   {
      printf("\t| %d |\n",stack[i]);
   }
}

void write_into_file()
{
          FILE *out_file;

   out_file = fopen("number.txt","w");

   if(out_file == NULL)
   {
      printf("error\n");
   }
   int i=0;
   for(i=top;i>=0;i--)
   {
      fprintf(out_file,"%d\n",stack[i]);
   }
   fclose(out_file);
}


void read_from_file()
{
   FILE *in_file;
   int n;
   in_file = fopen("number.txt","r");

   if(in_file == NULL)
   {
      printf("error\n");
   }
   while(fscanf(in_file,"%d",&n) != EOF)
   {
     push(n);
   }
   fclose(in_file);
}

void main()
{
   read_from_file();
   int ch,data;
   printf("1. PUSH\n");
   printf("2. POP\n");
   printf("3. DISPLAY\n");
   printf("4. EXIT\n");
   do{
      printf("Enter Your Choice : ");
      scanf("%d",&ch);
      switch(ch)
      {
         case 1:
            printf("Enter Data : ");
            scanf("%d",&data);
            push(data);
            break;
         case 2:
            pop();
            write_into_file();
            break;
         case 3:
            display();
            break;
         case 4:
            break;
         default :
            printf("Enter Proper Choice \n");
            break;
```

```
        }
    }while(ch!=4);
}
```

- OUTPUT



```
1. PUSH
2. POP
3. DISPLAY
4. EXIT
Enter Your Choice : 1
Enter Data : 7
Enter Your Choice : 2
         7 : DELETE
Enter Your Choice : 3
        | 2 |
        | 37 |
        | 50 |
        | 49 |
        | 27 |
        | 10 |
        | 2 |
        | 5 |
Enter Your Choice : 4

Process returned 4 (0x4)   execution time : 8.094 s
Press any key to continue.
```

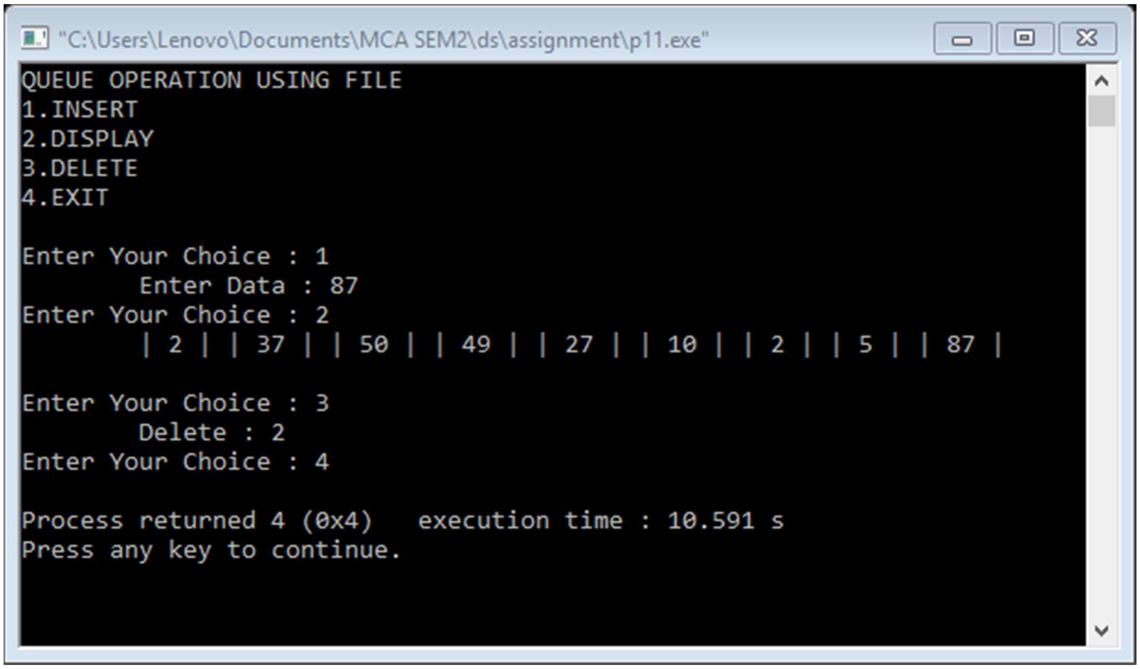
11. Write a program to implement Queue with required operations using array.

➢ PROGRAM

```
#include<stdio.h>
#define SIZE 100
int queue[SIZE];
int front=-1;
int rear=-1;

void insert(int item)
{
    if(rear > SIZE)
        printf("\tQueue is Overflow  : \n");
    else
    {
        if (front == - 1)
            front = 0;

        rear++;
```

```
      queue[rear]=item;
      write_into_file();
   }
}
void Delete()
{
   if(front > rear)
      printf("\tQueue is Underflow : \n");
   else
   {
      printf("\tDelete : %d\n",queue[front]);
      front++;
   }
}
void Display()
{
   int i;
   if(rear == -1 || front > rear)
      printf("\tQueue is Empty \n");
   printf("\t");
   for(i=front;i<=rear;i++)
      printf("| %d | ",queue[i]);
   printf("\n\n");
}
void write_into_file()
{
          FILE *out_file;

   out_file = fopen("number.txt","w");

   if(out_file == NULL)
   {
      printf("error\n");
   }
   int i=0;
   for(i=front;i<=rear;i++)
   {
      fprintf(out_file,"%d\n",queue[i]);
   }
   fclose(out_file);
}

void read_from_file()
{
   FILE *in_file;
   int n;

   in_file = fopen("number.txt","r");

   if(in_file == NULL)
   {
      printf("error\n");
   }
   while(fscanf(in_file,"%d",&n) != EOF)
   {
     insert(n);
   }
   fclose(in_file);
}
void main()
{
   read_from_file();
   printf("QUEUE OPERATION USING FILE\n");
   printf("1.INSERT\n");
   printf("2.DISPLAY\n");
   printf("3.DELETE\n");
   printf("4.EXIT\n\n");
   int ch;
   do{
      printf("Enter Your Choice : " );
      scanf("%d",&ch);
      switch(ch)
      {
         case 1:
            {
               int d;
               printf("\tEnter Data : ");
               scanf("%d",&d);
               insert(d);
               break;
            }
         case 2:
            Display();
            break;
         case 3:
            Delete();
            write_into_file();
            break;
         case 4:
            break;
         default:
            printf("\tEnter Proper Choice :\n");
            break;
      }
   }while(ch != 4);
}
```

- OUTPUT



```
QUEUE OPERATION USING FILE
1.INSERT
2.DISPLAY
3.DELETE
4.EXIT

Enter Your Choice : 1
        Enter Data : 87
Enter Your Choice : 2
        | 2 | | 37 | | 50 | | 49 | | 27 | | 10 | | 2 | | 5 | | 87 |

Enter Your Choice : 3
        Delete : 2
Enter Your Choice : 4

Process returned 4 (0x4)   execution time : 10.591 s
Press any key to continue.
```

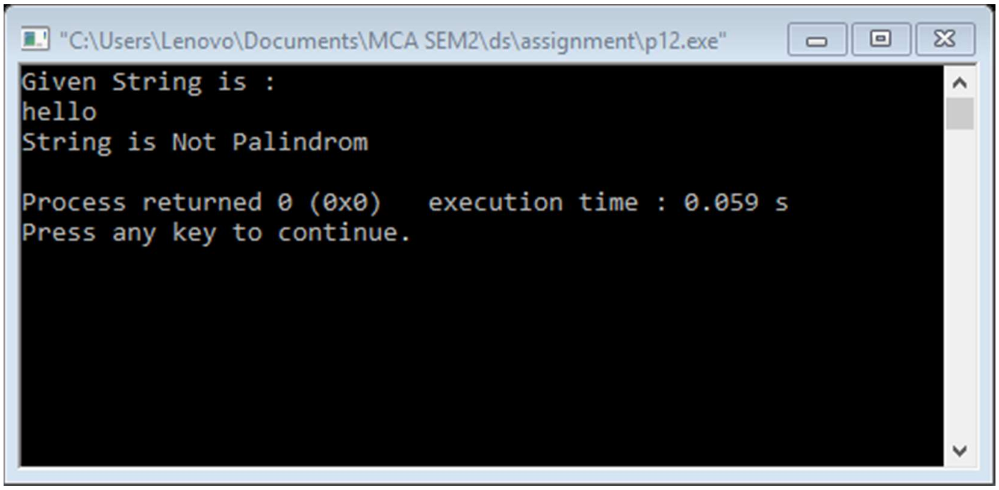
12. Write a program to check whether the string is palindrome or not. Use Stack Data Structure for the same.

➢ PROGRAM

```
#include<stdio.h>
#include<string.h>
#define SIZE 100
int stack[SIZE];
int top=-1;
char str[20];
void push(char c)
{
   if(top > SIZE)
      printf("Stack is OverFlow \n");
   else
   {
      top++;
      stack[top]=c;
   }
}
char pop()
{
   if(top == -1)
      printf("Stack is Underflow \n");
   else
   {
      char x=stack[top];
      top--;
      return x;
   }
}

void read_from_file()
{
   FILE *in_file;
   int n;

   in_file = fopen("string.txt","r");

   if(in_file == NULL)
   {
      printf("error\n");
   }
   int i=0;
   printf("Given String is :\n");
   while(fscanf(in_file,"%c",&n) != EOF)
   {
     printf("%c",n);
     str[i]=n;
     push(n);
     i++;
   }
   fclose(in_file);
}
int palindrom()
{
   char pal[20];
   int j=0;
   while(top != -1)
   {
      pal[j]=pop();
```

```
        j++;
    }
    if(strcmp(str,pal) == 0)
        return 1;
    else
        return 0;
}
void main()
{
    read_from_file();
    printf("\n");
    int res;
    res = palindrom();
    if(res == 1)
        printf("String is Palindrom \n");
    else
        printf("String is Not Palindrom \n");
}
```

- OUTPUT



```
Given String is :
hello
String is Not Palindrom

Process returned 0 (0x0)   execution time : 0.059 s
Press any key to continue.
```

13. Write a program to implement Doubly Linked List.

- PROGRAM

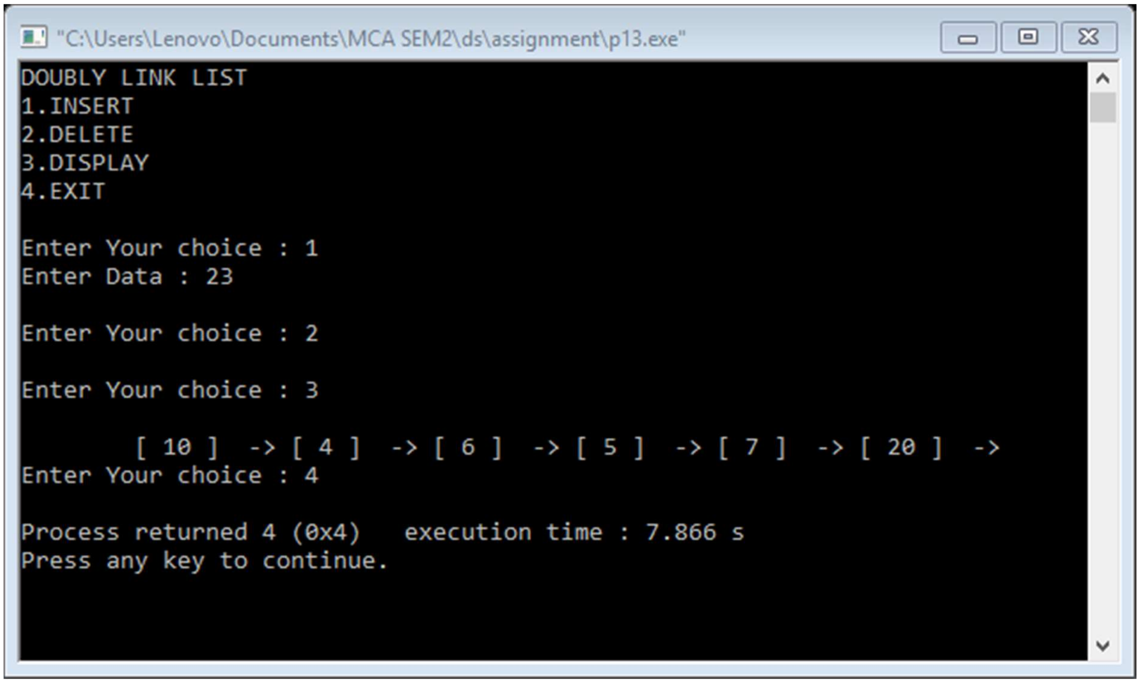
```
#include<stdio.h>
#include<stdlib.h>
struct node{
    int data;
    struct node *prev;
    struct node *next;
}*head=NULL;

void insert(int data)
```

```
{
   struct node *newnode,*temp;
   newnode=(struct node*)malloc(sizeof(struct node));
   newnode->data=data;
   newnode->next=NULL;
   newnode->prev=NULL;
   if(head == NULL)
   {
      head=newnode;
      newnode->next=NULL;
      newnode->prev=head;
   }
   else
   {
      temp=head;
      while(temp->next != NULL)
      {
         temp=temp->next;
      }
      temp->next=newnode;
      newnode->next=NULL;
      newnode->prev=temp;
   }
}

void Delete()
{
   struct node *temp;
   if(head == NULL)
   {
      printf("LINK LIST EMPTY\n");
   }
   else
   {
      temp=head;
      while(temp->next->next != NULL)
      {
         temp=temp->next;
      }
      temp->next=NULL;
   }
}

void write_into_file()
```

```
{
          FILE *out_file;

   out_file = fopen("Link_list.txt","w");

   if(out_file == NULL)
   {
      printf("error\n");
   }

   struct node *temp;
   temp=head;
   while(temp != NULL)
   {
      fprintf(out_file,"%d\n",temp->data);
      temp=temp->next;
   }
   fclose(out_file);
}

void read_from_file()
{
   FILE *in_file;
   int n;

   in_file = fopen("Link_list.txt","r");

   if(in_file == NULL)
   {
      printf("error\n");
   }
   while(fscanf(in_file,"%d",&n) != EOF)
   {
     insert(n);
   }
   fclose(in_file);
}

void Display()
{
   struct node *temp;
   temp=head;
   while(temp != NULL)
   {
      printf("[ %d ]  -> ",temp->data);
      temp=temp->next;
   }
}

void main()
{
   read_from_file();
   int ch,data,ser;
   printf("DOUBLY LINK LIST\n");
   printf("1.INSERT\n");
   printf("2.DELETE\n");
   printf("3.DISPLAY\n");
   printf("4.EXIT\n");
   do{
      printf("\nEnter Your choice : ");
      scanf("%d",&ch);
      switch(ch)
      {
         case 1:
            {
               printf("Enter Data : ");
               scanf("%d",&data);
               insert(data);
               write_into_file();
               break;
            }
         case 2:
            {
               Delete();
               write_into_file();
               break;
            }
         case 3:
            printf("\n\t");
            Display();
            break;
         case 4:
            return;
         default:
            printf("\tEnter Proper Choice : \n");
      }

   }while(ch != 4);
```

```
}
```

- OUTPUT



```
DOUBLY LINK LIST
1.INSERT
2.DELETE
3.DISPLAY
4.EXIT

Enter Your choice : 1
Enter Data : 23

Enter Your choice : 2

Enter Your choice : 3

        [ 10 ]  -> [ 4 ]  -> [ 6 ]  -> [ 5 ]  -> [ 7 ]  -> [ 20 ]  ->
Enter Your choice : 4

Process returned 4 (0x4)   execution time : 7.866 s
Press any key to continue.
```