PRINCIPAL OF PROGRAMMING LAB

(ETCS – 458)

Faculty name: Ms. ***Zameer Fatima***

Student name: Chander Jindal

Roll No.: 06514802719

Semester: 8th Semester

Group: 8C4

Maharaja Agrasen Institute of Technology

PSP Area, Sector – 22, Rohini, New Delhi – 110085

MAHARAJA AGRASEN INSTITUTE OF TECHNOLOGY

VISION

To nurture young minds in a learning environment of high academic value and imbibe spiritual and ethical values with technological and management competence.

MISSION

The Institute shall endeavor to incorporate the following basic missions in the teaching methodology:

Engineering Hardware – Software Symbiosis
Practical exercises in all Engineering and Management disciplines shall be carried out by Hardware equipment as well as the related software enabling deeper understanding of basic concepts and encouraging inquisitive nature.

Life – Long Learning
The Institute strives to match technological advancements and encourage students to keep updating their knowledge for enhancing their skills and inculcating their habit of continuous learning.

Liberalization and Globalization
The Institute endeavors to enhance technical and management skills of students so that they are intellectually capable and competent professionals with Industrial Aptitude to face the challenges of globalization.

Diversification
The Engineering, Technology and Management disciplines have diverse fields of studies with different attributes. The aim is to create a synergy of the above attributes by encouraging analytical thinking.

Entrepreneurship
The Institute strives to develop potential Engineers and Managers by enhancing their skills and research capabilities so that they become successful entrepreneurs and responsible citizens.

MAHARAJA AGRASEN INSTITUTE OF TECHNOLOGY

COMPUTER SCIENCE AND ENGINEERING DEPARTMENT

VISION

To produce "Critical Thinkers of Innovative Technology".

MISSION

To foster an open, multidisciplinary and highly collaborative research environment for producing world-class engineers capable of providing innovative solutions to real life problems and fulfil societal needs.

PRACTICAL RECORD

PAPER CODE	**:**	**ETCS-458**
Name of the student	**:**	**Chander Jindal**
University Roll No.	**:**	**06514802719**
Branch	**:**	**CSE**
Group	**:**	**8C-9**

PRACTICAL DETAILS

A) Experiments according to PPL lab syllabus prescribed by GGSIPU

Exp No.	Experiment Name	Date Of Performance	Total Marks	Signature with Date
1.	To implement all major functions of string.h in single C program using switch case to select specific function from user choice (like strlen, strcat, strcpy, strcmp, strrev).			
2.	Write a program in C to reverse a linked list iterative and recursive.			
3.	Write a program in C to implement iterative Towers of Hanoi.			
4.	WAP in C++ to count the numbers of object of a class with the help of static data member, function and constructor.			
6.	WAP in C++ to define a class Complex to represents set of all complex numbers. Overload '+' operator to add two complex numbers using member function of the class and overload '*' operator to multiply two complex numbers using friend function of the class complex.			
8.	Write a program in to prepare a list of 50 questions and their answers.			
9.	Write a program to display 10 questions at random out of exp. 8-50 questions.			
10.	Implement producer-consumer problem using threads.			

Experiment-1

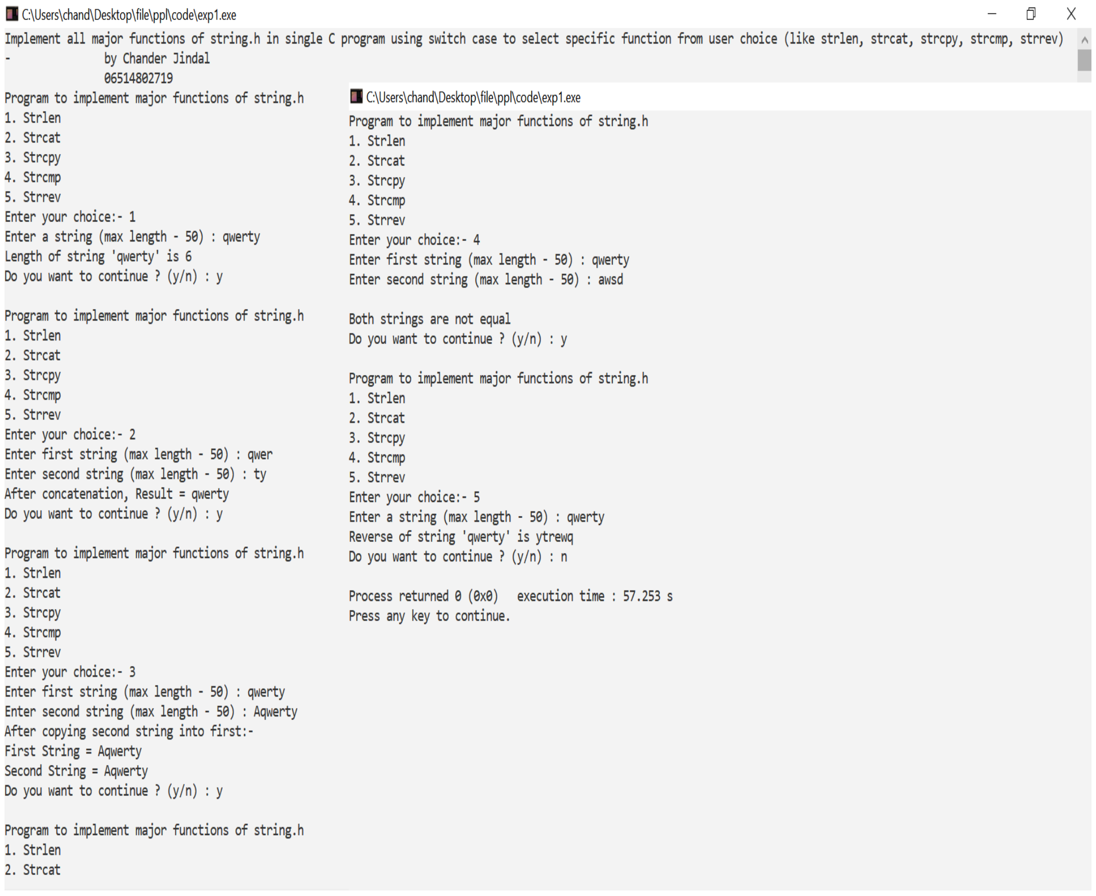
Aim: To implement all major functions of string.h in single C program using switch case to select specific function from user choice (like strlen, strcat, strcpy, strcmp, strrev).

Code:

```
#include <stdio.h>
#include <string.h>
//#include <conio.h>
int main(){
   printf("Implement all major functions of string.h in single C program using switch case to select
specific function from user choice (like strlen, strcat, strcpy, strcmp, strrev)\n-\t\tby Chander Jindal\n\t\
t06514802719");
   int ch;
   char s1[50], s2[50], c;
   do
   {
      //clrscr();
      printf("\nProgram to implement major functions of string.h");
      printf("\n1. Strlen\n2. Strcat\n3. Strcpy\n4. Strcmp\n5. Strrev");
      printf("\nEnter your choice:- ");
      scanf("%d", &ch);
      fflush(stdin);
      switch (ch)
      {
      case 1:
         printf("Enter a string (max length - 50) : ");
         gets(s1);
         printf("Length of string '%s' is %d", s1, strlen(s1));
         break;
      case 2:
         printf("Enter first string (max length - 50) : ");
         gets(s1);
         printf("Enter second string (max length - 50) : ");
         gets(s2);
         printf("After concatenation, Result = %s", strcat(s1, s2));
         break;
      case 3:
         printf("Enter first string (max length - 50) : ");
         gets(s1);
         printf("Enter second string (max length - 50) : ");
         gets(s2);
         printf("After copying second string into first:-");
         strcpy(s1, s2);
         printf("\nFirst String = %s", s1);
         printf("\nSecond String = %s", s2);
```

```
            break;
        case 4:
            printf("Enter first string (max length - 50) : ");
            gets(s1);
            printf("Enter second string (max length - 50) : ");
            gets(s2);
            if (strcmp(s1, s2) == 0)
            {
                printf("\nBoth strings are equal");
            }
            else
            {
                printf("\nBoth strings are not equal");
                break;
            }
        case 5:
            printf("Enter a string (max length - 50) : ");
            gets(s1);
            char s[50];
            strcpy(s, s1);
            printf("Reverse of string '%s' is %s", s, strrev(s1));
            break;
        default:
            printf("\n Wrong choice");
            break;
        }
        printf("\nDo you want to continue ? (y/n) : ");
        scanf("%c", &c);
    } while ((c == 'y') || (c == 'Y'));
    return 0;
}
```

Output:



```
C:\Users\chand\Desktop\file\ppl\code\exp1.exe
Implement all major functions of string.h in single C program using switch case to select specific function from user choice (like strlen, strcat, strcpy, strcmp, strrev)
-               by Chander Jindal
                06514802719
Program to implement major functions of string.h
1. Strlen
2. Strcat
3. Strcpy
4. Strcmp
5. Strrev
Enter your choice:- 1
Enter a string (max length - 50) : qwerty
Length of string 'qwerty' is 6
Do you want to continue ? (y/n) : y

Program to implement major functions of string.h
1. Strlen
2. Strcat
3. Strcpy
4. Strcmp
5. Strrev
Enter your choice:- 2
Enter first string (max length - 50) : qwer
Enter second string (max length - 50) : ty
After concatenation, Result = qwerty
Do you want to continue ? (y/n) : y

Program to implement major functions of string.h
1. Strlen
2. Strcat
3. Strcpy
4. Strcmp
5. Strrev
Enter your choice:- 3
Enter first string (max length - 50) : qwerty
Enter second string (max length - 50) : Aqwerty
After copying second string into first:-
First String = Aqwerty
Second String = Aqwerty
Do you want to continue ? (y/n) : y

Program to implement major functions of string.h
1. Strlen
2. Strcat
```

```
C:\Users\chand\Desktop\file\ppl\code\exp1.exe
Program to implement major functions of string.h
1. Strlen
2. Strcat
3. Strcpy
4. Strcmp
5. Strrev
Enter your choice:- 4
Enter first string (max length - 50) : qwerty
Enter second string (max length - 50) : awsd

Both strings are not equal
Do you want to continue ? (y/n) : y

Program to implement major functions of string.h
1. Strlen
2. Strcat
3. Strcpy
4. Strcmp
5. Strrev
Enter your choice:- 5
Enter a string (max length - 50) : qwerty
Reverse of string 'qwerty' is ytrewq
Do you want to continue ? (y/n) : n

Process returned 0 (0x0)   execution time : 57.253 s
Press any key to continue.
```

VIVA QUESTIONS

Q. What is the use of string.h header while and where is this file stored?

⇒ String.h is the header file required for string functions. This function appends not more than n characters from the string pointed to by src to the end of the string pointed to by dest plus a terminating Null-character. It is stored in "/usr/include".

Q. How can we create a header file?

⇒ To make a header file, we have to create one file with a name, and extension should be (*.h). In that function there will be no main() function. In that file, we can put some variables, some functions etc. To use that header file, it should be present at the same directory, where the program is located. Now using #include we have to put the header file name. The name will be inside double quotes. Include syntax will be look like this. #include ”header_file.h”.

Q. Write the function to find the length of a string.

⇒
```
int StrLength(char[] str){
 int i;
 for(i = 0; Str[i] != '\0'; ++i);
 return i;  }
```

Q. Write the function to concatenate two strings.

⇒
```
#include <stdio.h> int main() {
char s1[100] = "programming ",
s2[] = "is awesome";
int length, j;
length = 0;
while (s1[length] != '\0') {     ++length;   }  // concatenate s2 to s1
for (j = 0; s2[j] != '\0'; ++j, ++length) {     s1[length] = s2[j];   }
s1[length] = '\0';
printf("After concatenation: ");
puts(s1);
 return 0; }
```

Q. Explain the use of header file.

⇒ A header file is a file with extension .h which contains C function declarations and macro definitions to be shared between several source files. There are two types of header files: the files that the programmer writes and the files that comes with your compiler.

Experiment-2

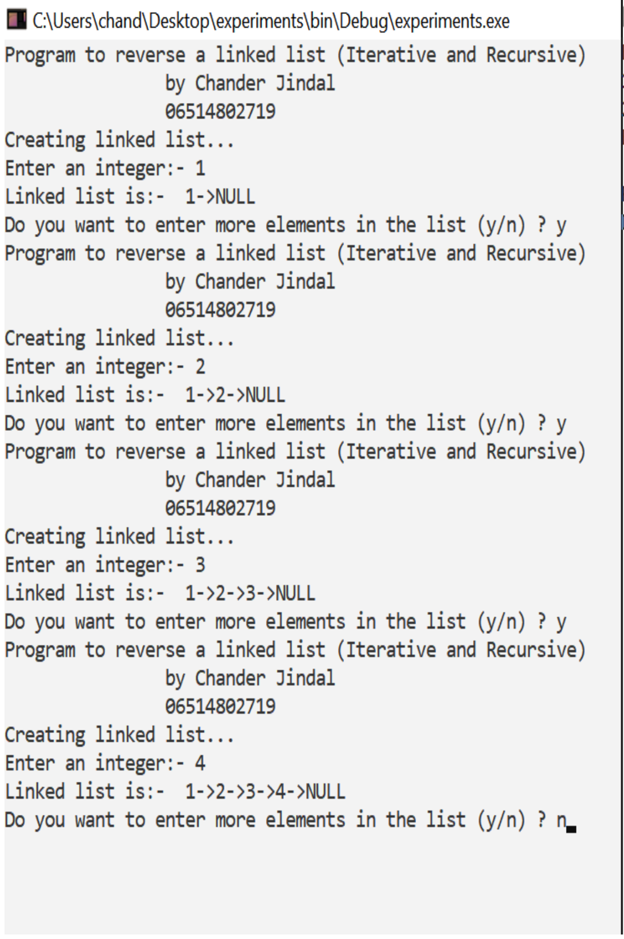
Aim: Write a program in C to reverse a linked list iterative and recursive.

Code

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

typedef struct node{
int i;
struct node *next; } node1;

void reverselinkedlist(node1 *head){
if(head->next==NULL)
printf("%d->",head->i);
else{
reverselinkedlist(head->next);
printf("%d->",head->i);
}}

int main(){
char hh;
do{
char ch;
node1 *head=NULL,*tail=NULL;
do{
printf("Program to reverse a linked list (Iterative and Recursive)\n\t\tby Chander Jindal\n\t\
t06514802719");
printf("\nCreating linked list...");
printf("\nEnter an integer:- ");
if(head==NULL){
head = malloc(sizeof(node1));
scanf("%d",&head->i);
head->next=NULL;
tail = head; }
else{
node1 *n1 = malloc(sizeof(node1));
scanf("%d",&n1->i);
n1->next=NULL;
tail->next=n1;
tail=n1; }
printf("Linked list is:- ");
node1 *n2=head;
printf(" ");
while(n2!=NULL){
printf("%d->",n2->i);
n2 = n2->next; }
printf("NULL");
printf("\nDo you want to enter more elements in the list (y/n) ? ");
fflush(stdin);
scanf("%c",&ch);
} while(ch=='y'||ch=='Y');
char ch1;
do{
int choice;
system("cls");
printf("Entered Linked List is:- ");
node1 *n2=head;
printf(" ");
while(n2!=NULL){
printf("%d->",n2->i);
n2 = n2->next; }
printf("NULL");
printf("\n1. Reverse a linked list iteratively");
printf("\n2. Reverse a linked list recursively");
printf("\nEnter your choice - ");
scanf("%d",&choice);
switch(choice){
case 1: printf(" ");
node1 *n2 = head;
node1 *c,*chead=NULL;
while(n2!=NULL){
c = malloc(sizeof(node1));
c->i = n2->i;
if(chead==NULL){
c->next = NULL;
chead = c; }
else{
c->next = chead;
chead = c;}
n2 = n2->next; }
node1 *x = chead;
printf("\nReversed linked list is : ");
while(x!=NULL){
printf("%d->",x->i);
x=x->next; }
printf("NULL"); break;
case 2:printf("\nReversed linked list is : ");
reverselinkedlist(head);
printf("NULL"); break;
default:printf("\nWrong choice"); break; }
printf("\nDo you want to reverse linked list again using a different choice (y/n)? ");
fflush(stdin);
scanf("%c",&ch1);
} while((ch1=='Y')||(ch1=='y'));
printf("\nDo you want to run program again (y/n)? ");
```

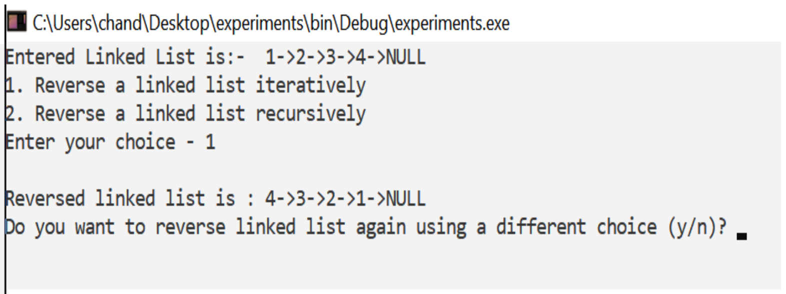
```
fflush(stdin);
scanf("%c",&hh);
} while((hh=='y')||(hh=='Y'));
return 0;
}
```

Output:



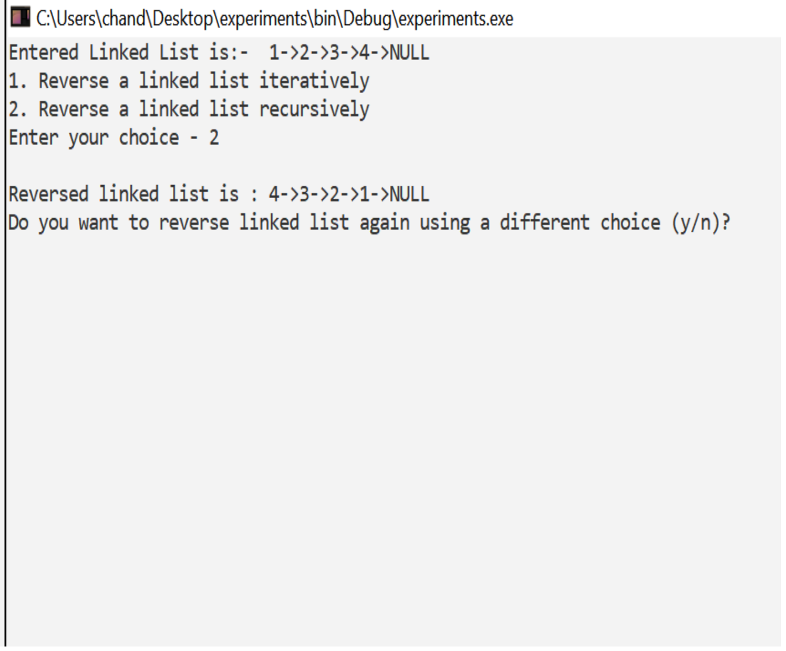
C:\Users\chand\Desktop\experiments\bin\Debug\experiments.exe

```
Program to reverse a linked list (Iterative and Recursive)
                by Chander Jindal
                06514802719
Creating linked list...
Enter an integer:- 1
Linked list is:-  1->NULL
Do you want to enter more elements in the list (y/n) ? y
Program to reverse a linked list (Iterative and Recursive)
                by Chander Jindal
                06514802719
Creating linked list...
Enter an integer:- 2
Linked list is:-  1->2->NULL
Do you want to enter more elements in the list (y/n) ? y
Program to reverse a linked list (Iterative and Recursive)
                by Chander Jindal
                06514802719
Creating linked list...
Enter an integer:- 3
Linked list is:-  1->2->3->NULL
Do you want to enter more elements in the list (y/n) ? y
Program to reverse a linked list (Iterative and Recursive)
                by Chander Jindal
                06514802719
Creating linked list...
Enter an integer:- 4
Linked list is:-  1->2->3->4->NULL
Do you want to enter more elements in the list (y/n) ? n
```



C:\Users\chand\Desktop\experiments\bin\Debug\experiments.exe

```
Entered Linked List is:-  1->2->3->4->NULL
1. Reverse a linked list iteratively
2. Reverse a linked list recursively
Enter your choice - 1

Reversed linked list is : 4->3->2->1->NULL
Do you want to reverse linked list again using a different choice (y/n)?
```



C:\Users\chand\Desktop\experiments\bin\Debug\experiments.exe

```
Entered Linked List is:-  1->2->3->4->NULL
1. Reverse a linked list iteratively
2. Reverse a linked list recursively
Enter your choice - 2

Reversed linked list is : 4->3->2->1->NULL
Do you want to reverse linked list again using a different choice (y/n)?
```

VIVA QUESTIONS

Q. What is the difference between iterative and recursive function call?

⇒ Recursion and iteration are both different ways to execute a set of instructions repeatedly. The main difference between these two is that in recursion, we use function calls to execute the statements repeatedly inside the function body, while in iteration, we use loops like "for" and "while" to do the same.

Q. What are formal parameters in functions?

⇒ Formal parameters are always variables, while actual parameters do not have to be variables. Parameter Written in Function Call is Called "Actual Parameter". One can use numbers, expressions, or even function calls as actual parameters.

Q. How is the structure node declared?

⇒
```
typedef struct node{
int value;
struct node *next;
} node;
node *createNode(int val){
node *newNode = malloc(sizeof(node));
newNode->value = val;
newNode->next = NULL;
return newNode;    }
```

Q. Define a link list.

⇒ A linked list is a linear data structure, in which the elements are not stored at contiguous memory locations. The elements in a linked list are linked using pointers.

Experiment-3

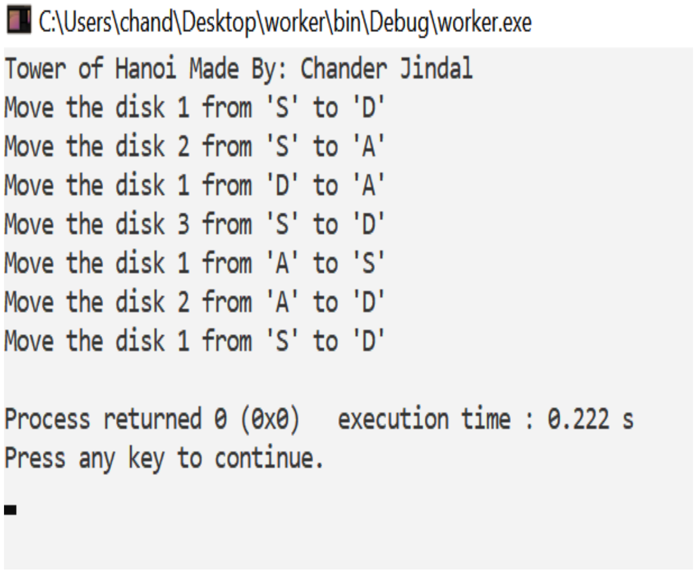
Aim: Write a program in C to implement iterative Towers of Hanoi.

Code

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

struct Stack{
unsigned capacity; int top;
int *array;
};

struct Stack* createStack(unsigned capacity) {
struct Stack* stack =
(struct Stack*) malloc(sizeof(struct Stack));
stack -> capacity = capacity;
stack -> top = -1;
stack -> array =
(int*) malloc(stack -> capacity * sizeof(int));
return stack; }
int isFull(struct Stack* stack){
return (stack->top == stack->capacity - 1); }
int isEmpty(struct Stack* stack) {
return (stack->top == -1); }
void push(struct Stack *stack, int item) {
if (isFull(stack)) return;
stack -> array[++stack -> top] = item; }
int pop(struct Stack* stack) {
if (isEmpty(stack))
return INT_MIN;
return stack -> array[stack -> top--]; }
void moveDisk(char fromPeg, char toPeg, int disk) { printf("Move the disk %d from \'%c\' to \'%c\'\n",
disk, fromPeg, toPeg); }
void moveDisksBetweenTwoPoles(struct Stack *src, struct Stack *dest, char s, char d) {
int pole1TopDisk = pop(src);
int pole2TopDisk = pop(dest);
if (pole1TopDisk == INT_MIN) {
push(src, pole2TopDisk);
moveDisk(d, s, pole2TopDisk); }
else if (pole2TopDisk == INT_MIN) {
push(dest, pole1TopDisk);
moveDisk(s, d, pole1TopDisk); }
else if (pole1TopDisk > pole2TopDisk) {
```

```
push(src, pole1TopDisk);
push(src, pole2TopDisk);
moveDisk(d, s, pole2TopDisk);
}else {
push(dest, pole2TopDisk);
push(dest, pole1TopDisk);
moveDisk(s, d, pole1TopDisk); }}
void tohIterative(int num_of_disks, struct Stack *src, struct Stack *aux, struct Stack *dest){
int i, total_num_of_moves;
char s = 'S', d = 'D', a = 'A';
if (num_of_disks % 2 == 0){
char temp = d; d = a; a = temp; }
total_num_of_moves = pow(2, num_of_disks) - 1;
for (i = num_of_disks; i >= 1; i--)
push(src, i);
for (i = 1; i <= total_num_of_moves; i++){
if (i % 3 == 1)
moveDisksBetweenTwoPoles(src, dest, s, d);
else if (i % 3 == 2)
moveDisksBetweenTwoPoles(src, aux, s, a);
else if (i % 3 == 0)
moveDisksBetweenTwoPoles(aux, dest, a, d);
}}
int main(){
unsigned num_of_disks = 3;
printf("Tower of Hanoi Made By: Chander Jindal\n");
struct Stack *src, *dest, *aux;
src = createStack(num_of_disks);
aux = createStack(num_of_disks);
dest = createStack(num_of_disks);
tohIterative(num_of_disks, src, aux, dest);
return 0;}
```

Output:



```
C:\Users\chand\Desktop\worker\bin\Debug\worker.exe
Tower of Hanoi Made By: Chander Jindal
Move the disk 1 from 'S' to 'D'
Move the disk 2 from 'S' to 'A'
Move the disk 1 from 'D' to 'A'
Move the disk 3 from 'S' to 'D'
Move the disk 1 from 'A' to 'S'
Move the disk 2 from 'A' to 'D'
Move the disk 1 from 'S' to 'D'

Process returned 0 (0x0)   execution time : 0.222 s
Press any key to continue.
```

VIVA QUESTIONS

Q. What is the tower of Hanoi problem?

⇒ Tower of Hanoi is a mathematical puzzle where we have three rods and n disks. The objective of the puzzle is to move the entire stack to another rod, obeying the following simple rules: Only one disk can be moved at a time.

Q. What are the various ways is stack created?

⇒ There are two ways to implement a stack: Using array. Using linked list.

Q. List the models of computation of language.

⇒ Models of computation can be classified into three categories: sequential models, functional models, and concurrent models.

Q. What are objectives of principle of programming language?

⇒ Objectives are-

- To introduce several different paradigms of programming
- To gain experience with these paradigms by using example programming languages
- To understand concepts of syntax, translation, abstraction, and implementation

Q. What are the Paradigms of Programming?

⇒ Paradigm can also be termed as method to solve some problem or do some task. Programming paradigm is an approach to solve problem using some programming language or also we can say it is a method to solve a problem using tools and techniques that are available to us following some approach. There are lots for programming language that are known but all of them need to follow some strategy when they are implemented and this methodology/strategy is paradigms. Apart from varieties of programming language there are lots of paradigms to fulfil each and every demand.

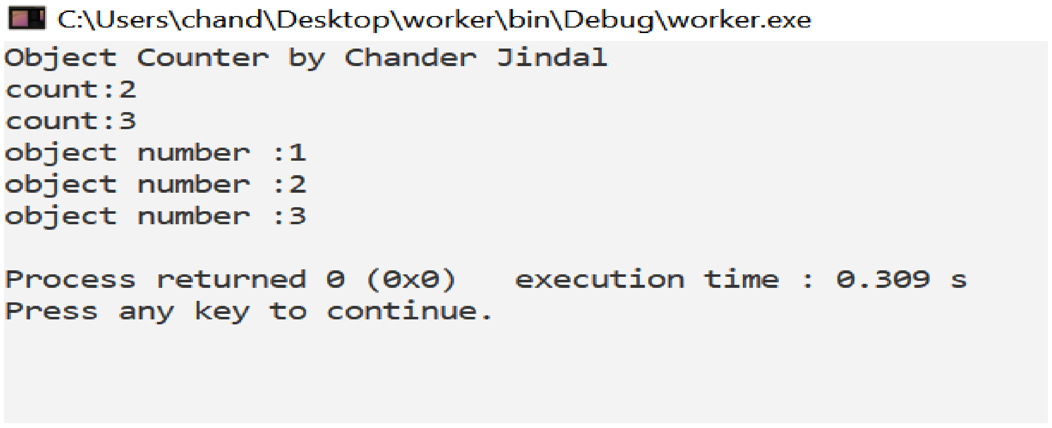
Experiment-4

Aim: WAP in C++ to count the numbers of object of a class with the help of static data member, function and constructor.

Code:

```
#include <iostream>
using namespace std;
class test{
int objNo;
static int objCnt;
public:
test(){
objNo = ++objCnt;}
~test(){
--objCnt; }
void printObjNumber(void){
cout << "object number :" << objNo << "\n";}
static void printObjCount(void){
cout << "count:" << objCnt<< "\n";
}};
int test::objCnt;
int main(){
cout<<"Object Counter by Chander Jindal\n";
test t1, t2;
test::printObjCount();
test t3;
test::printObjCount();
t1.printObjNumber();
t2.printObjNumber();
t3.printObjNumber();
return 0; }
```

Output:



```
C:\Users\chand\Desktop\worker\bin\Debug\worker.exe
Object Counter by Chander Jindal
count:2
count:3
object number :1
object number :2
object number :3

Process returned 0 (0x0)   execution time : 0.309 s
Press any key to continue.
```

VIVA QUESTIONS

Q. List various type of languages.

⇒ 1. Procedural Programming Language
2. Object Oriented Programming Language
3. Parallel Processing Programming Language
4. Logic Programming Language
5. Functional Programming Language
6. Database Programming Language

Q. What are the issues for languages?

⇒ 1. Development on hardware technologies.
2. Simplicity and performance trade off.
3. Usage Area
4. Portability
5. Variety of Project's Design Patterns
6. Safety and security

Q. What is translation?

⇒ Translation is the process of conversion of source code into the low level machine code.

Q. What are different types of translation and their roles?

⇒ Compiler
A compiler is a translator used to convert high-level programming language to low-level programming language. It converts the whole program in one session and reports errors detected after the conversion.

⇒ Interpreter
Just like a compiler, is a translator used to convert high-level programming language to low-level programming language. It converts the program one at a time and reports errors detected at once while doing the conversion.

⇒ Assembler
An assembler is is a translator used to translate assembly language to machine language. It is like a compiler for the assembly language but interactive like an interpreter.

Q. What is trade's off of translation?

⇒ Trade's off of translation are-
• Compilation – lower level machine may be faster, so programs run faster – compilation can be expensive – examples: C
• Interpretation – more ability to perform diagnostics (or changes) at run-time – examples: Basic, UNIX shells, Lisp

Experiment-5

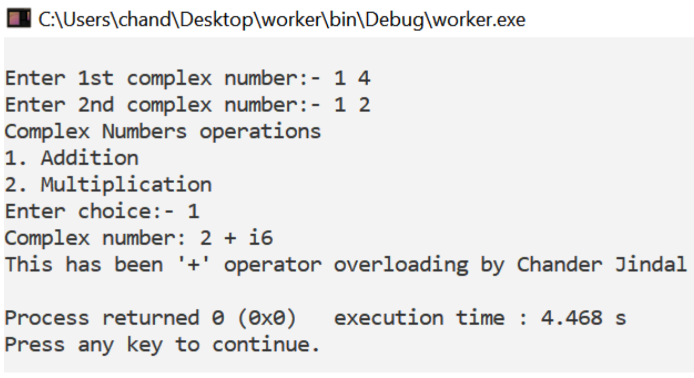
Aim: WAP in C++ to define a class Complex to represents set of all complex numbers. Overload '+' operator to add two complex numbers using member function of the class and overload '*' operator to multiply two complex numbers using friend function of the class complex.

Code:

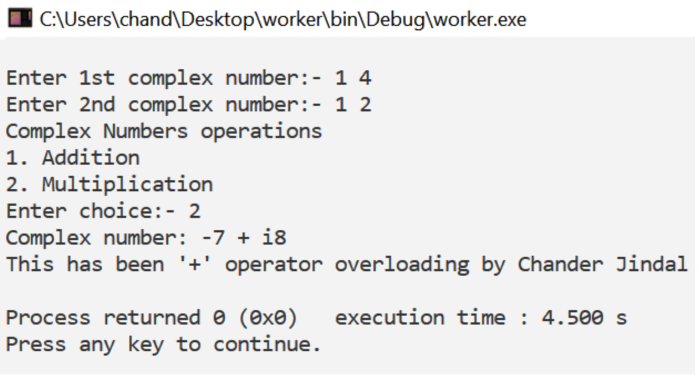
```
#include <iostream>
using namespace std;
class Complex{
private: float real;
float imag;
public:
Complex(): real(0), imag(0){ }
void input(){ int a[2];
for(int i=0; i<2; i++){ cin>>a[i]; }
real = a[0]; imag = a[1]; }
Complex operator + (Complex c){
Complex temp;
temp.real = real + c.real;
temp.imag = imag + c.imag;
return temp; }
friend Complex operator * (Complex c1, Complex c2);
void output(){
if(imag < 0) cout << "Complex number: "<< real<< "i" << imag ;
else cout << "Complex number: " <<real << " + i" <<imag; }};
Complex operator * (Complex c1, Complex c2) {
Complex c3;
c3.real=c1.real*c2.real-c1.imag*c2.imag;
c3.imag=c1.imag*c2.imag;
return(c3); }
int main(){
Complex b, c, result;
int ch;
cout<<"\nEnter 1st complex number:- ";b.input();
cout<<"Enter 2nd complex number:- ";c.input();
cout<<"Complex Numbers operations\n1. Addition\n2. Multiplication";
cout<<"\nEnter choice:- ";cin>>ch;
switch(ch){
case 1:result = b + c;
result.output(); break;
case 2:result = b*c;
result.output(); break; }
cout<<"\nThis has been '+' operator overloading by Chander Jindal\n";
```

return 0;}

Output:



```
C:\Users\chand\Desktop\worker\bin\Debug\worker.exe

Enter 1st complex number:- 1 4
Enter 2nd complex number:- 1 2
Complex Numbers operations
1. Addition
2. Multiplication
Enter choice:- 1
Complex number: 2 + i6
This has been '+' operator overloading by Chander Jindal

Process returned 0 (0x0)   execution time : 4.468 s
Press any key to continue.
```



```
C:\Users\chand\Desktop\worker\bin\Debug\worker.exe

Enter 1st complex number:- 1 4
Enter 2nd complex number:- 1 2
Complex Numbers operations
1. Addition
2. Multiplication
Enter choice:- 2
Complex number: -7 + i8
This has been '+' operator overloading by Chander Jindal

Process returned 0 (0x0)   execution time : 4.500 s
Press any key to continue.
```

VIVA QUESTIONS

Q. List the benefits of modular development approach.

⇒ 1. Consistency
2. Reduced Development times
3. Flexibility

Q. Give some reasons why computer scientists and professional software developers should study general concepts of language design and evaluation.

⇒ A solid programming languages foundation enables students to effectively recognize when designing a new language is appropriate and how to avoid these problems. Understanding programming language principles and models often provides the in- sights leading to new innovations as well.

Q. What constitutes a programming environment?

⇒ File system, Text Editor, Linker, Compiler, Integrated tools.

Q. Give an example of how aliasing deters reliability.

⇒ A pointer is a value that represents a memory address sometimes 2 pointers can represent the same memory address thats what aliasing is
int * p;
*p = 5;
int * alias;
alias = p;
the variable alias is an alias of p and *alias is equal to 5 if you change *alias then *p changes along with it.

Q. How do type declaration statements effect the readability of programming language?

⇒ Readability is a key element in any programming language and thus, very important. The addition of type declarations helps to improve code readability. This is because it makes it easy to identify and differentiate different variables by data types. A program with well-defined type declarations is easy to understand.

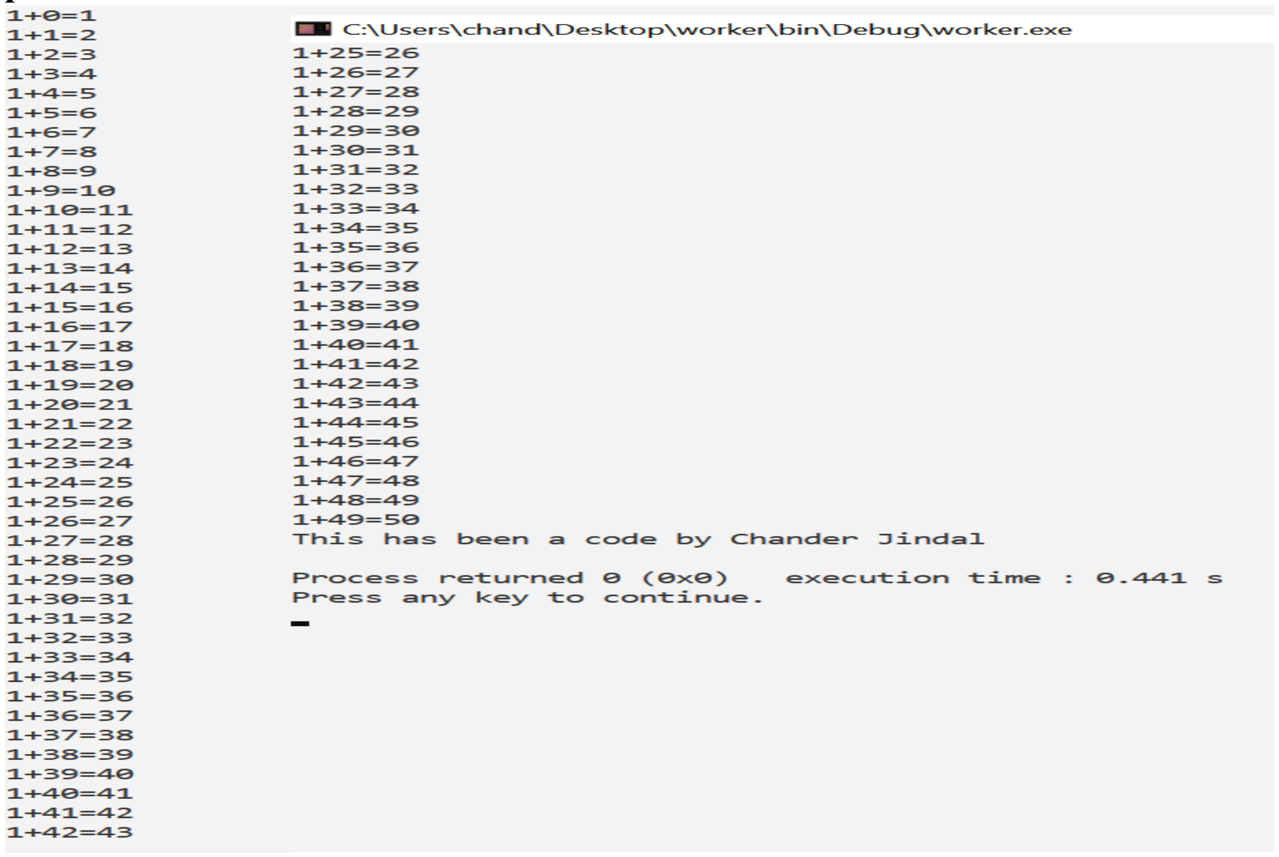
Experiment-6

Aim: Write a program in to prepare a list of 50 questions and their answers.

Code:

```
#include <iostream>
#include <string>
#include <sstream>
using namespace std;
int main() {
string ques[50],ans[50];
for(int i=0,j=1;i<50;i++,j++){
stringstream s1,s2;
s1 << i; s2 << j;
ques[i]="1+";
ques[i]+=s1.str();
ans[i]=s2.str();}
for(int i=0;i<50;i++) cout<<ques[i]<<"="<<ans[i]<<endl;
cout<<"This has been a code by Chander Jindal"<<endl;
return 0; }
```

Output:



```
1+0=1
1+1=2
1+2=3
1+3=4
1+4=5
1+5=6
1+6=7
1+7=8
1+8=9
1+9=10
1+10=11
1+11=12
1+12=13
1+13=14
1+14=15
1+15=16
1+16=17
1+17=18
1+18=19
1+19=20
1+20=21
1+21=22
1+22=23
1+23=24
1+24=25
1+25=26
1+26=27
1+27=28
1+28=29
1+29=30
1+30=31
1+31=32
1+32=33
1+33=34
1+34=35
1+35=36
1+36=37
1+37=38
1+38=39
1+39=40
1+40=41
1+41=42
1+42=43
```

```
C:\Users\chand\Desktop\worker\bin\Debug\worker.exe
1+25=26
1+26=27
1+27=28
1+28=29
1+29=30
1+30=31
1+31=32
1+32=33
1+33=34
1+34=35
1+35=36
1+36=37
1+37=38
1+38=39
1+39=40
1+40=41
1+41=42
1+42=43
1+43=44
1+44=45
1+45=46
1+46=47
1+47=48
1+48=49
1+49=50
This has been a code by Chander Jindal

Process returned 0 (0x0)   execution time : 0.441 s
Press any key to continue.
```

VIVA QUESTIONS

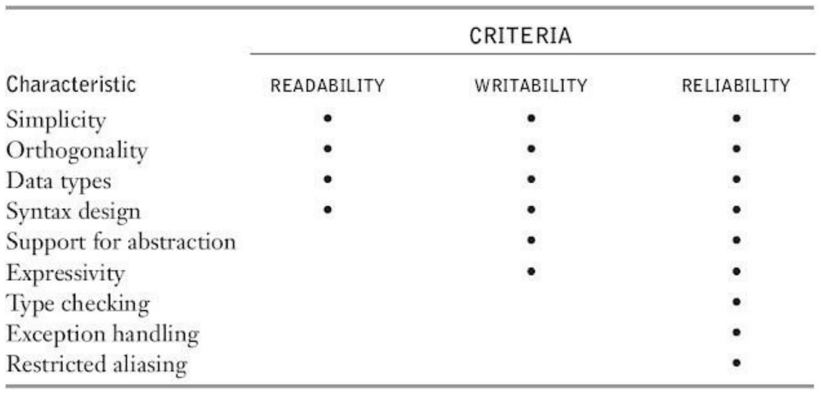
Q. Explain language evaluation criteria and the characteristics that affect them.

⇒ The most prominent language evaluation criteria are:

⇒ Readability

⇒ Writability

⇒ Reliability



	CRITERIA		
Characteristic	READABILITY	WRITABILITY	RELIABILITY
Simplicity	•	•	•
Orthogonality	•	•	•
Data types	•	•	•
Syntax design	•	•	•
Support for abstraction		•	•
Expressivity		•	•
Type checking			•
Exception handling			•
Restricted aliasing			•

Q. What is the use of functions? How are actual parameters different from formal parameters?

⇒ In computer science, Backus–Naur form or Backus normal form (BNF) is a notation technique for context-free grammars, often used to describe the syntax of languages used in computing, such as computer programming languages, document formats, instruction sets and communication protocols. They are applied wherever exact descriptions of languages are needed: for instance, in official language specifications, in manuals, and in textbooks on programming language theory.

Q. What is a void pointer?

⇒ A void pointer is a pointer that has no associated data type with it. A void pointer can hold address of any type and can be typcasted to any type.

-

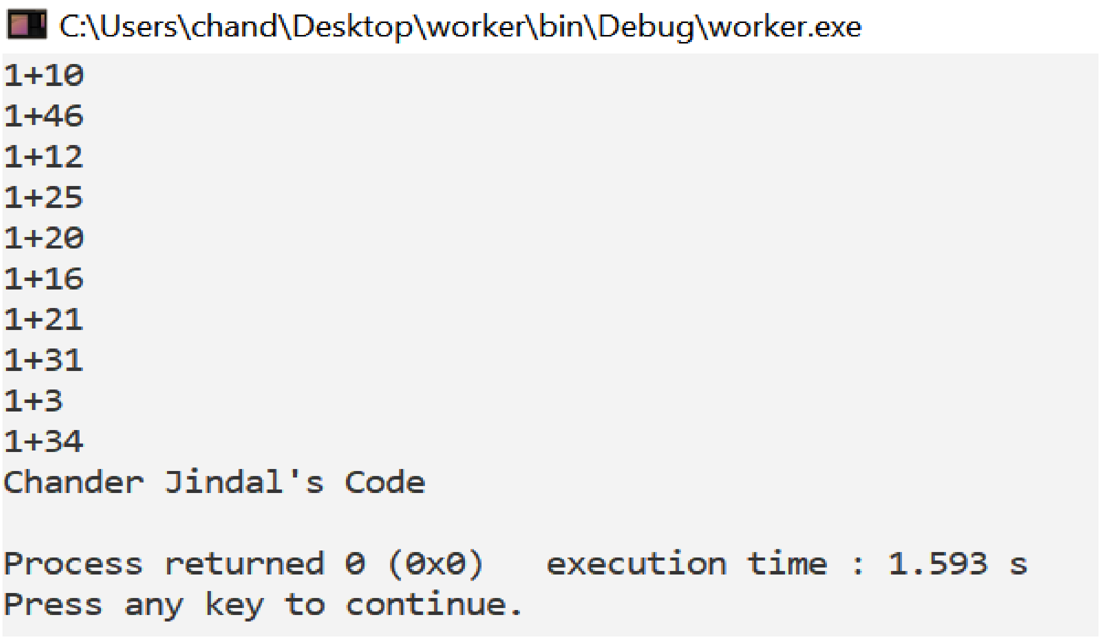
Experiment-7

Aim: Write a program to display 10 questions at random out of exp.8-50 questions (do not display the answers of these questions to the user now).

Code:

```
#include <iostream>
#include <string>
#include <sstream>
#include <ctime>
#include <stdlib.h>
using namespace std;

void generateSetOfNumbers(int arr[]) {
int p[50] =
{0,1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,22,23,24,25,26,27,28,29,30,31,32,33,34,35,36,
37,38,39,40,41,42,43,44,45,46,47,48,49};
for (int i=49; i>0; --i){
int j = rand()%i;
int temp = p[i];
p[i] = p[j];
p[j] = temp; }
for (int i=0; i<10; ++i)
arr[i] = p[i]; }
int main(){
string ques[50],ans[50]; int r[10];
for(int i=0,j=1;i<50;i++,j++){
stringstream s1,s2;
s1 << i; s2 << j;
ques[i]="1+";
ques[i]+=s1.str();
ans[i]=s2.str();
}
srand(time(0)); generateSetOfNumbers(r); for(int i=0;i<10;i++){
int x=r[i]; cout<<ques[x]<<endl; }
cout<<"Chander Jindal's Code\n";
return 0;}
```

Output:



```
C:\Users\chand\Desktop\worker\bin\Debug\worker.exe
1+10
1+46
1+12
1+25
1+20
1+16
1+21
1+31
1+3
1+34
Chander Jindal's Code

Process returned 0 (0x0)   execution time : 1.593 s
Press any key to continue.
```

VIVA QUESTIONS

Q. List various types of pointers.

⇒ NULL pointer   Dangling pointer
⇒ Generic pointer   Wild pointer
⇒ Complex pointer

Q. What is the use of functions? How are actual parameters different from formal parameters?

⇒ Functions provide a high degree of modularity for your application and also provide better code reusability and ease of maintenance.
⇒ Actual parameters are used in function calling statement. Formal parameters are used in function definition statement.

Q. How are threads created?

⇒ In Call by value method original value is not modified whereas, in Call by reference method, the original value is modified.
⇒ In Call by value, a copy of the variable is passed whereas in Call by reference, a variable itself is passed.
⇒ In Call by value, actual and formal arguments will be created in different memory locations whereas in Call by reference, actual and formal arguments will be created in the same memory location.

Experiment-8

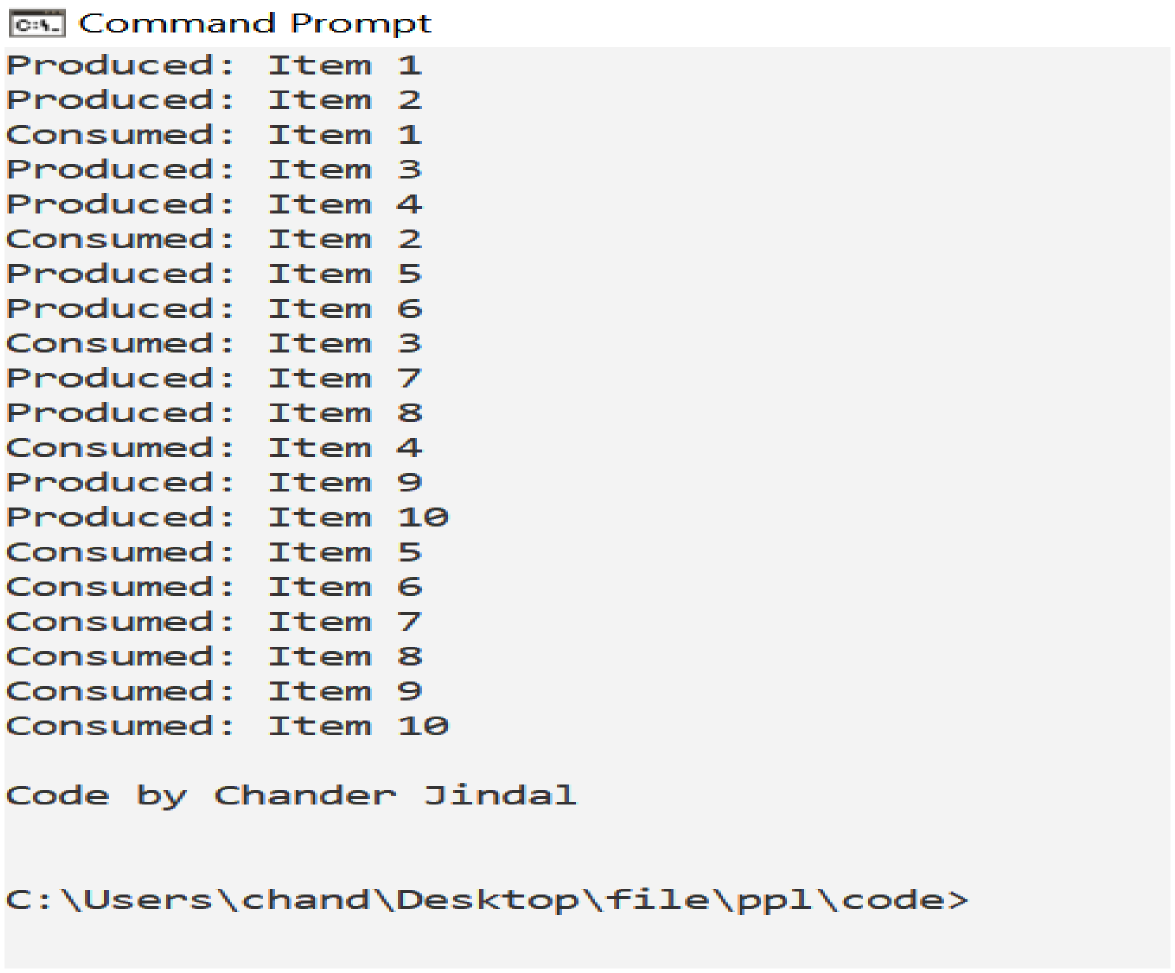
Aim: Implement producer-consumer problem using threads.

Code:

```
#include<bits/stdc++.h>
#include<pthread.h>
#include<semaphore.h>
#include <unistd.h>
using namespace std;

int r1,total_produced=0,total_consume=0;

sem_t notEmpty;

void* produce(void *arg){
   while(1){
   cout<<"Producer produces item."<<endl;
   cout<<"Total produced = "<<++total_produced<<
      " Total consume = "<<total_consume*-1<<endl;
   sem_post(¬Empty);
   sleep(rand()%100*0.01);
   }
}

void* consume(void *arg){
   while(1){
   sem_wait(¬Empty);
   cout<<"Consumer consumes item."<<endl;
   cout<<"Total produced = "<<total_produced<<
      " Total consume = "<<(--total_consume)*-1<<endl;
   sleep(rand()%100*0.01);
   }
}

int main(int argv,char *argc[]){

   pthread_t producer,consumer;
   pthread_attr_t attr;

   sem_init(¬Empty,0,0);

   pthread_attr_init(&attr);
   pthread_attr_setdetachstate(&attr,PTHREAD_CREATE_JOINABLE);

   r1=pthread_create(&producer,&attr,produce,NULL);
   if(r1){
   cout<<"Error in creating thread"<<endl;
   exit(-1);
   }
```

```
  r1=pthread_create(&consumer,&attr,consume,NULL);
  if(r1){
  cout<<"Error in creating thread"<<endl;
  exit(-1);
  }

  pthread_attr_destroy(&attr);

  r1=pthread_join(producer,NULL);
  if(r1){
  cout<<"Error in joining thread"<<endl;
  exit(-1);
  }
  r1=pthread_join(consumer,NULL);
  if(r1){
  cout<<"Error in joining thread"<<endl;
  exit(-1);
  }
  pthread_exit(NULL);
  cout<<"\nCode by Chander Jindal\n";
  return 0;
}
```

Output:



```
Produced: Item 1
Produced: Item 2
Consumed: Item 1
Produced: Item 3
Produced: Item 4
Consumed: Item 2
Produced: Item 5
Produced: Item 6
Consumed: Item 3
Produced: Item 7
Produced: Item 8
Consumed: Item 4
Produced: Item 9
Produced: Item 10
Consumed: Item 5
Consumed: Item 6
Consumed: Item 7
Consumed: Item 8
Consumed: Item 9
Consumed: Item 10

Code by Chander Jindal


C:\Users\chand\Desktop\file\ppl\code>
```

VIVA QUESTIONS

Q. What is the role of producer and consumer in producer-consumer problem?

⇒ We have a buffer of fixed size. A producer can produce an item and place it in the buffer. A consumer can pick items and consume them. We need to ensure that when a producer is placing an item in the buffer, then at the same time consumer should not consume any item. In this problem, buffer is the critical section.

Q. What is a semaphore?

⇒ Semaphore is simply a variable that is non-negative and shared between threads. A semaphore is a signaling mechanism, and a thread that is waiting on a semaphore can be signaled by another thread. It uses two atomic operations, 1) wait, and 2) signal for the process synchronization.

⇒ A semaphore either allows or disallows access to the resource, which depends on how it is set up.

Q. How are threads created?

⇒ Java lets you create threads in one of two ways:

⇒ By implementing the Runnable Interface

⇒ By extending the Thread class