PRINCIPLE OF PROGRAMMING LANGUAGES LAB

ETCS-458

Faculty Name: Mrs. Savita Sharma

Name: Prince

Roll No.: 10314802720

Semester: 8th

Group: 8C6

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 endeavour 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 endeavour's 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.

Digitization of Learning Processes

The Institute provides seamless opportunities for innovative learning in all Engineering and Management disciplines through digitization of learning processes using analysis, synthesis, simulation, graphics, tutorials and related tools to create a platform for multi-disciplinary approach.

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 & ENGINEERING DEPARTMENT

VISION

To Produce “Critical thinkers of Innovative Technology”

MISSION

To provide an excellent learning environment across the computer sciencediscipline to inculcate professional behaviour, strong ethical values, innovative research capabilities and leadership abilities which enable them to become successful entrepreneurs in this globalized world.

1. To nurture an **excellent learning environment** that helps students to enhance their problem-solving skills and to prepare students to be lifelong learners by offering a solid theoretical foundation with applied computing experiences and educating them about their **professional, and ethical responsibilities**.
2. To establish **Industry-Institute Interaction**, making students ready for the industrial environment and be successful in their professional lives.
3. To promote **research activities** in the emerging areas of technology convergence.
4. To build engineers who can look into technical aspects of an engineering solution thereby setting a ground for producing successful **entrepreneur.**

PRINCIPILES OF PROGRAMMING LANGUAGES LAB

PRACTICAL RECORD

PAPER CODE	**:**	**ETCS - 458**
Name of the student	:	Prince
University Roll No.	:	10314802720
Branch	:	CSE
Group	:	8C6

PRACTICAL DETAILS

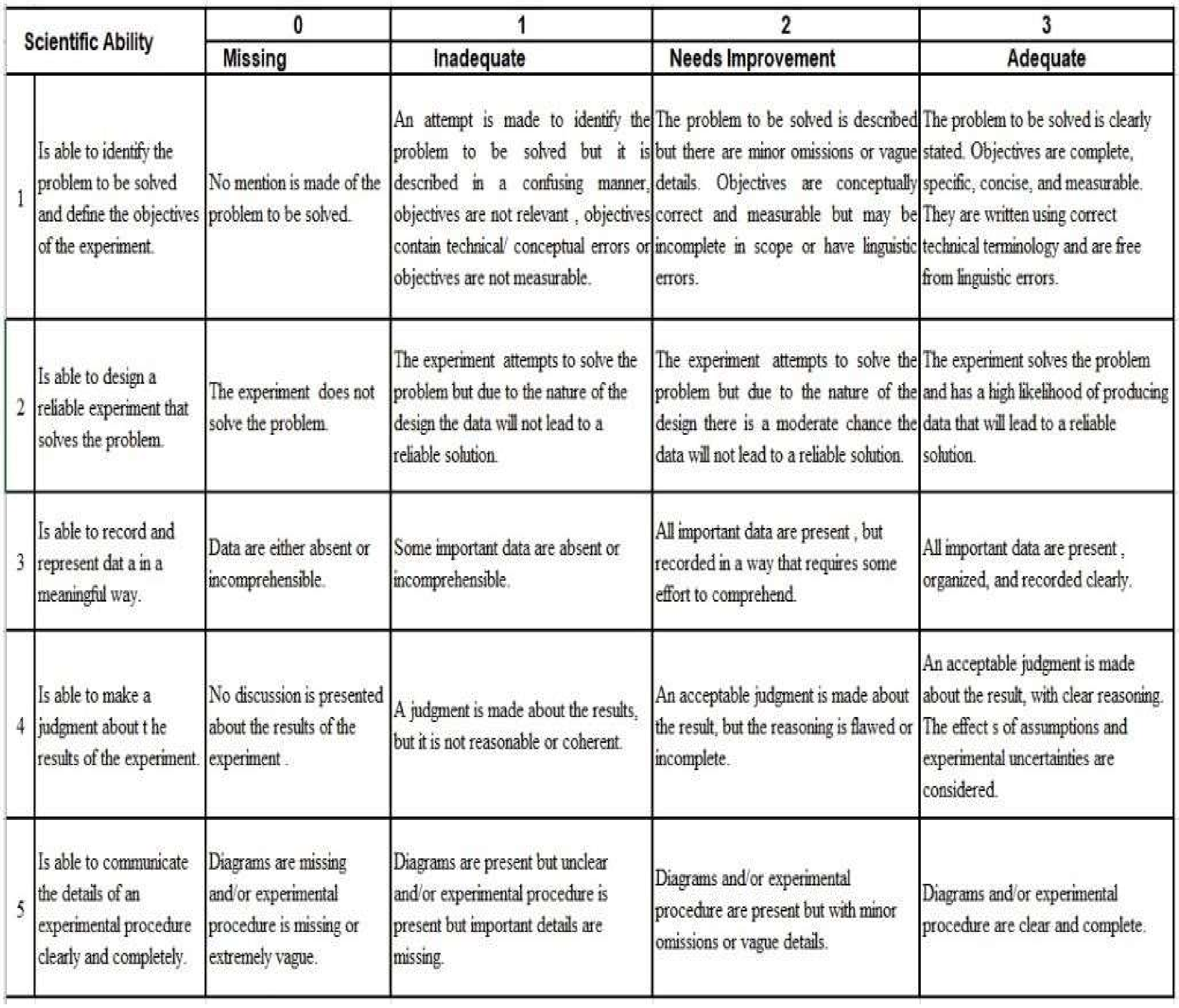
a) List of experiments

Exp. no	**Experiment Name**	**Date of performance**	**Date of checking**	**Marks**	**Signature**
1.	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 (WAP) in C toreverse a linked list iterative and recursive.				
3.	WAP in C to implement iterative Towers of Hanoi				
4.	WAP in C++ to count thenos. of object of a class withthe help of static data member, function andconstructor.				
5.	WAP in C++ & Java to declare a class Time with data members mm for minutes, ss for seconds and				

	hh for hours. Define a parameterize constructor to assign time to its objects. Add two-time objects using member function and assign to third objects. Implement all possible cases of time.				
6.	WAP in C++ to define a class Complex to represents set of all complex numbers. Overload '+' operator toadd two complex numbers using member function of the class and overload '*' operator to multiply two complex numbers using friend function of the class complex.				
7.	Implement simple multi-threaded server to perform all mathematics operation parallel inJava.				
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 (do not display the answer of these questions to the user now).				

10.	Implement producer-consumer problem using threads.				
11.	There are 200 questions on a 3hr examination. Amongthese questions are 50 mathematics problems. It is suggested that twice asmuch time be spent on each maths problem as for eachother question. WAP which calculates how many minutes should be spent on mathematics problems.				
12.	Write a Program where it may or may not print counter value in sequence and every time we run it, it produces a different result based on CPU availability to a thread.				
13.	Two polynomials are entered by the user in the form of : ax2 + bx + c where the powers of x can be any integer value and a,b& c are constants. Now WAP in C and JAVA which calculates the sum, product and difference of the two polynomials.				
14.	In June a baseball team that played 60 games had won 30% of its game played. After a phenomenal winning streak this team raised its average to 50% . WAP which calculates how many games must the team have won in a row to attain this average.				
15.	A company contracts to paint 3 houses. Mr. Brown can paint a house in 6 days while Mr. Black would take 8 days and Mr. Blue 12 days. After 8 days Mr. Brown goes on vacation and Mr. Black begins to work for a period of 6 days. Write a program in java which calculates how days will it take Mr. Blue to complete the contract.				

Rubrics



Scientific Ability		0	1	2	3
		Missing	Inadequate	Needs Improvement	Adequate
1	Is able to identify the problem to be solved and define the objectives of the experiment.	No mention is made of the problem to be solved.	An attempt is made to identify the problem to be solved but it is described in a confusing manner, objectives are not relevant , objectives contain technical/ conceptual errors or objectives are not measurable.	The problem to be solved is described but there are minor omissions or vague details. Objectives are conceptually correct and measurable but may be incomplete in scope or have linguistic errors.	The problem to be solved is clearly stated. Objectives are complete, specific, concise, and measurable. They are written using correct technical terminology and are free from linguistic errors.
2	Is able to design a reliable experiment that solves the problem.	The experiment does not solve the problem.	The experiment attempts to solve the problem but due to the nature of the design the data will not lead to a reliable solution.	The experiment attempts to solve the problem but due to the nature of the design there is a moderate chance the data will not lead to a reliable solution.	The experiment solves the problem and has a high likelihood of producing data that will lead to a reliable solution.
3	Is able to record and represent dat a in a meaningful way.	Data are either absent or incomprehensible.	Some important data are absent or incomprehensible.	All important data are present , but recorded in a way that requires some effort to comprehend.	All important data are present , organized, and recorded clearly.
4	Is able to make a judgment about t he results of the experiment.	No discussion is presented about the results of the experiment .	A judgment is made about the results, but it is not reasonable or coherent.	An acceptable judgment is made about the result, but the reasoning is flawed or incomplete.	An acceptable judgment is made about the result, with clear reasoning. The effect s of assumptions and experimental uncertainties are considered.
5	Is able to communicate the details of an experimental procedure clearly and completely.	Diagrams are missing and/or experimental procedure is missing or extremely vague.	Diagrams are present but unclear and/or experimental procedure is present but important details are missing.	Diagrams and/or experimental procedure are present but with minor omissions or vague details.	Diagrams and/or experimental procedure are clear and complete.

Experiment 1

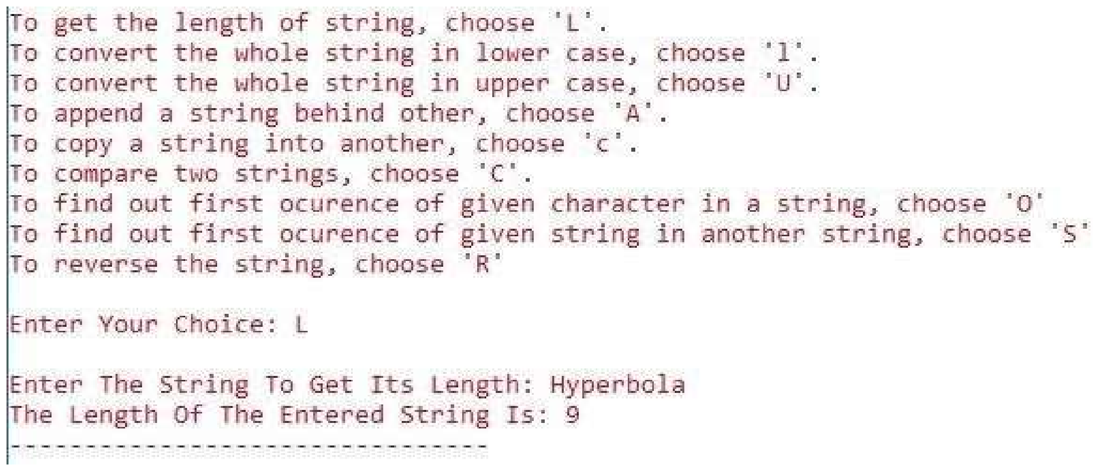
AIM: - Implement all major functions of string.h in single C program using switch case to select specific function from user choice (like strlen, strcat, strcpy, strcmp, strrev)

CODE: -

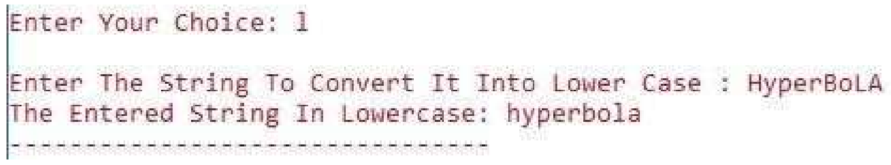
```
#include<stdio.h>
#include<string.h>
void main(){
	char str[50];
	char temp[20];
	char choice, ch;
	//printf("For ")
	puts("To get the length of string, choose 'L'.");		//strlen();//
	puts("To convert the whole string in lower case, choose 'l'.");		//strlwr();
	puts("To convert the whole string in upper case, choose 'U'."); //strupr();
	puts("To append a string behind other, choose 'A'.");//strcat();//
	puts("To copy a string into another, choose 'c'.");	//strcpy();//
	puts("To compare two strings, choose 'C'."); //strcmp();//
	puts("To find out first ocurence of given character in a string, choose 'O'"); //strchr();
	puts("To find out first ocurence of given string in another string, choose 'S'");
	//strstr();
	puts("To reverse the string, choose 'R'");	//strrev();//
	printf("\nEnter Your Choice: ");
	scanf("%c", &choice);
	switch(choice){
		case 'L':
			printf("\nEnter The String To Get Its Length: ");
			scanf("%s", str);
			printf("The Length Of The Entered String Is: %d", strlen(str));
			break;
		case 'l':
			printf("\nEnter The String To Convert It Into Lower Case : ");
			scanf("%s", str);
			printf("The Entered String In Lowercase: %s", strlwr(str));
			break;
		case 'U':
			printf("\nEnter The String To Convert It Into Upper Case : ");
			scanf("%s", str);
			printf("\nThe Entered String In Lowercase: %s", strupr(str));
			break;
		case 'A':
			printf("\nEnter The First String: ");
			scanf("%s", str);
			printf("Enter The Second String To Append It Behind First One: ");
			scanf("%s", temp);
			strcat(str, temp);
			printf("\nNow, The First String Is: %s", str);
```

```
                        break;
                case 'c':
                        printf("\nEnter The First String: ");
                        scanf("%s", str);
                        printf("Enter The Second String: ");
                        scanf("%s", temp);
                        strcpy(str, temp);
                        printf("\nNow, The First String Is: %s", str);
                        printf("\nAnd, The Second String Is: %s", temp);
                        break;
                case 'C':
                        printf("\nEnter The First String: ");
                        scanf("%s", str);
                        printf("Enter The Second String: ");
                        scanf("%s", temp);
                        if(strcmp(str, temp)==0)printf("\nBoth Strings Are Similar.");
                        else printf("\nBoth Strings Are Different.");
                        break;
                case 'O':
                        printf("\nEnter The String: ");
                        scanf("%s", str);
                        printf("Enter The Character To Be Searched: ");
                        scanf("%c", &ch);
                        printf("\nThe First Occurence of Character Is At: %s", strchr(str, ch));
                        break;
                case 'S':
                        printf("\nEnter The String: ");
                        scanf("%s", str);
                        printf("Enter The String To Be Searched: ");
                        scanf("%s", temp);
                        printf("\nThe First Occurence of Character Is At: %s", strstr(str,
temp));
                        break;
                case 'R':
                        printf("\nEnter The String To Get Its Reverse: ");
                        scanf("%s", str);
                        printf("\nThe Reverse Of The Entered String Is: %s", strrev(str));
                        break;
                default:
                        printf("\nYou Entered A Wrong Choise.");
                        break;
        }


}
```

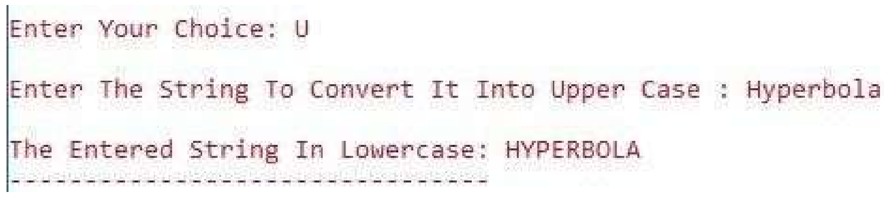
OUTPUT



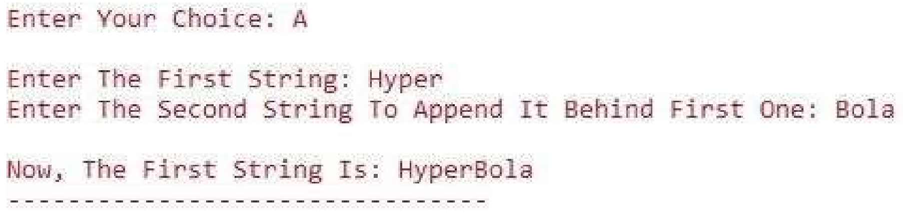
```
To get the length of string, choose 'L'.
To convert the whole string in lower case, choose 'l'.
To convert the whole string in upper case, choose 'U'.
To append a string behind other, choose 'A'.
To copy a string into another, choose 'c'.
To compare two strings, choose 'C'.
To find out first ocurence of given character in a string, choose 'O'
To find out first ocurence of given string in another string, choose 'S'
To reverse the string, choose 'R'

Enter Your Choice: L

Enter The String To Get Its Length: Hyperbola
The Length Of The Entered String Is: 9
--------------------------------
```



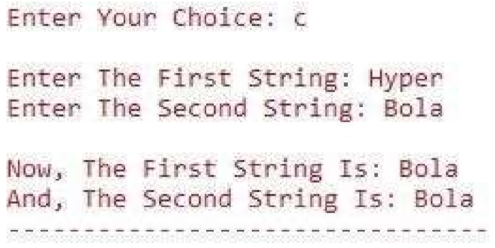
```
Enter Your Choice: l

Enter The String To Convert It Into Lower Case : HyperBoLA
The Entered String In Lowercase: hyperbola
--------------------------------
```



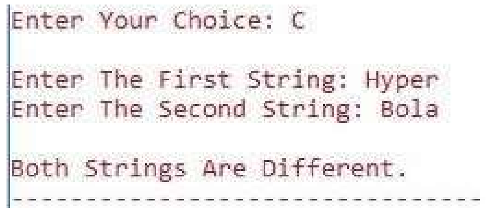
```
Enter Your Choice: U

Enter The String To Convert It Into Upper Case : Hyperbola

The Entered String In Lowercase: HYPERBOLA
--------------------------------
```



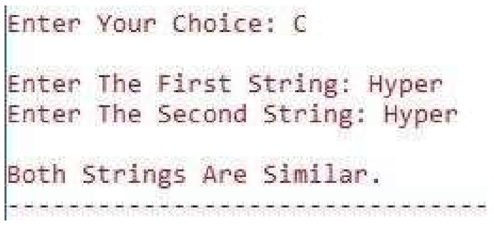
```
Enter Your Choice: A

Enter The First String: Hyper
Enter The Second String To Append It Behind First One: Bola

Now, The First String Is: HyperBola
--------------------------------
```



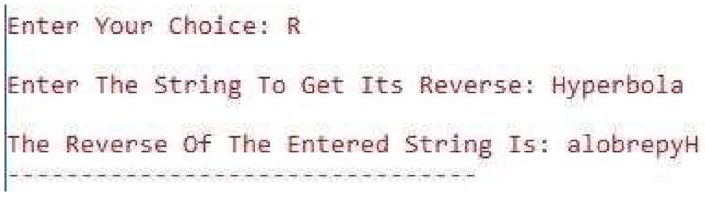
```
Enter Your Choice: c

Enter The First String: Hyper
Enter The Second String: Bola

Now, The First String Is: Bola
And, The Second String Is: Bola
--------------------------------
```



```
Enter Your Choice: C

Enter The First String: Hyper
Enter The Second String: Bola

Both Strings Are Different.
--------------------------------
```



```
Enter Your Choice: C

Enter The First String: Hyper
Enter The Second String: Hyper

Both Strings Are Similar.
--------------------------------
```



```
Enter Your Choice: R

Enter The String To Get Its Reverse: Hyperbola

The Reverse Of The Entered String Is: alobrepyH
-------------------------------
```

VIVA QUESTIONS

Q1. What is the use of the string.h header file and where is it stored?
A1. The string.h header file is used for string manipulation functions in C programming. It is a standard library header file stored in the C library.

Q2. How can we create a header file?
A2. To create a header file, you can simply create a new text file and save it with a .h extension. Write the necessary function prototypes, definitions, and declarations in the header file.

Q3. Write the function to find the length of a string.
A3. The function to find the length of a string in C can be written as:

```
int string_length(const char* str) {
int length = 0;
while (str[length] != '\0') {
length++;
}
return length;
}
```

Q4. Write the function to concatenate two strings.
A4. The function to concatenate two strings in C can be written as:

```
void string_concatenate(char* destination, const char* source) {
int dest_len = string_length(destination);
int i;
for (i = 0; source[i] != '\0'; i++) {
destination[dest_len + i] = source[i];
}
destination[dest_len + i] = '\0';
}
```

Q5. Explain the use of a header file.
A5. Header files in programming are used to provide function prototypes, type definitions, and macro definitions to other source code files. They allow for code reusability, modularization, and help in organizing and managing complex projects.

Experiment 2

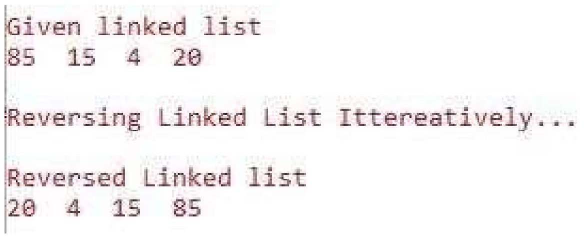
AIM: - Write a program (WAP) in C to reverse a linked list iterative and recursive.

CODE: -

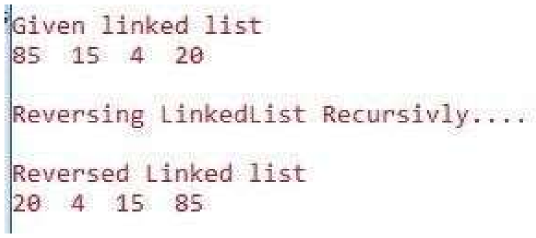
```
// Iterative C program to reverse a linked list
#include<stdio.h>
#include<stdlib.h>

/* Link list node */
struct Node
{
   int data;
   struct Node* next;
};

/* Function to reverse the linked list */
void recursiveReverse(struct Node* head, struct Node** headRef)
{
        struct Node* first;
        struct Node* rest;
        // empty list base case
        if (head == NULL)
           return;

        first = head;                  // suppose first = {1, 2, 3}
        rest = first->next;        // rest = {2, 3}

        // base case: List has only one node
        if (rest == NULL)
        {
                // fix the head pointer here
                *headRef = first;
                 return;
        }

        // Recursively reverse the smaller {2, 3} case
        // after: rest = {3, 2}
        recursiveReverse(rest, headRef);

        // put the first elem on the end of the list
        rest->next = first;
        first->next = NULL;  // (tricky step -- make a drawing)
}
void reverse(struct Node** head_ref)
{
   struct Node* prev  = NULL;
   struct Node* current = *head_ref;
   struct Node* next = NULL;
```

```
    printf("\n\nReversing Linked List Ittereatively...\n");
    while (current != NULL)
    {
        // Store next
        next  = current->next;

        // Reverse current node's pointer
        current->next = prev;

        // Move pointers one position ahead.
        prev = current;
        current = next;
    }
    *head_ref = prev;
}

/* Function to push a node */
void push(struct Node** head_ref, int new_data)
{
    struct Node* new_node =
            (struct Node*) malloc(sizeof(struct Node));
    new_node->data = new_data;
    new_node->next = (*head_ref);
    (*head_ref)    = new_node;
}

/* Function to print linked list */
void printList(struct Node *head)
{
    struct Node *temp = head;
    while(temp != NULL)
    {
        printf("%d ", temp->data);
        temp = temp->next;
    }
}

int main()
{
    /* Start with the empty list */
    struct Node* head = NULL;
    push(&head, 20);
     push(&head, 4);
     push(&head, 15);
     push(&head, 85);

     printf("Given linked list\n");
     printList(head);

    //reverse(&head);
    printf("\n\nReversing LinkedList Recursivly .. \n");
            recursiveReverse(head, &head);

             printf("\nReversed Linked list \n");
     printList(head);
     getchar();
}
```

OUTPUT: -



```
Given linked list
85  15  4  20

Reversing Linked List Ittereatively...

Reversed Linked list
20  4  15  85
```



```
Given linked list
85  15  4  20

Reversing LinkedList Recursivly....

Reversed Linked list
20  4  15  85
```

VIVA QUESTIONS

Q1. What is the difference between iterative and recursive function call?
A1. Iterative function calls use loops to repeatedly execute a block of code, while recursive function calls refer to a function calling itself to solve a problem by breaking it down into smaller subproblems.

Q2. What are formal parameters in functions?
A2. Formal parameters in functions are variables used to represent the values that will be passed into the function when it is called, serving as placeholders for the actual arguments.

Q3. How is the structure node declared?
A3. The structure node is declared by defining a user-defined data type that contains data elements and a pointer to the next node in the linked list.

Q4. Define a linked list.
A4. A linked list is a data structure composed of nodes where each node contains a data element and a reference (or link) to the next node, forming a sequence of connected nodes.

Q5. Why do we need to store the address of the starting node of a linked list for reversing a list?
A5. The address of the starting node is needed to reverse a linked list because it acts as the entry point for traversing and manipulating the list, allowing us to rearrange the connections between nodes and update the references accordingly.

Experiment 3

AIM: - WAP in C to implement iterative Towers of Hanoi

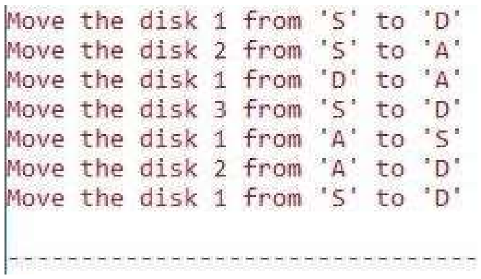
CODE: -

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

// A structure to represent a stack
struct Stack
{
unsigned capacity;
int top;
int *array;
};

// function to create a stack of given capacity.
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;
}

// Stack is full when top is equal to the last index
int isFull(struct Stack* stack)
{
return (stack->top == stack->capacity - 1);
}

// Stack is empty when top is equal to -1
int isEmpty(struct Stack* stack)
{
return (stack->top == -1);
}

// Function to add an item to stack. It increases
// top by 1
void push(struct Stack *stack, int item)
{
   if (isFull(stack))
      return;
   stack -> array[++stack -> top] = item;
```

```
}

// Function to remove an item from stack. It
// decreases top by 1
int pop(struct Stack* stack)
{
   if (isEmpty(stack))
      return INT_MIN;
   return stack -> array[stack -> top--];
}

//Function to show the movement of disks
void moveDisk(char fromPeg, char toPeg, int disk)
{
   printf("Move the disk %d from \'%c\' to \'%c\'\n",
      disk, fromPeg, toPeg);
}

// Function to implement legal movement between
// two poles
void moveDisksBetweenTwoPoles(struct Stack *src,
         struct Stack *dest, char s, char d)
{
   int pole1TopDisk = pop(src);
   int pole2TopDisk = pop(dest);

   // When pole 1 is empty
   if (pole1TopDisk == INT_MIN)
   {
      push(src, pole2TopDisk);
      moveDisk(d, s, pole2TopDisk);
   }

   // When pole2 pole is empty
   else if (pole2TopDisk == INT_MIN)
   {
      push(dest, pole1TopDisk);
      moveDisk(s, d, pole1TopDisk);
   }

   // When top disk of pole1 > top disk of pole2
   else if (pole1TopDisk > pole2TopDisk)
   {
      push(src, pole1TopDisk);
      push(src, pole2TopDisk);
      moveDisk(d, s, pole2TopDisk);
   }

   // When top disk of pole1 < top disk of pole2
   else
```

```
    {
        push(dest, pole2TopDisk);
        push(dest, pole1TopDisk);
        moveDisk(s, d, pole1TopDisk);
    }
}

//Function to implement TOH puzzle
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 number of disks is even, then interchange
    //destination pole and auxiliary pole
    if (num_of_disks % 2 == 0)
    {
        char temp = d;
        d = a;
        a = temp;
    }
    total_num_of_moves = pow(2, num_of_disks) - 1;

    //Larger disks will be pushed first
    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);
    }
}

// Driver Program
int main()
{
    // Input: number of disks
    unsigned num_of_disks = 3;

    struct Stack *src, *dest, *aux;
    // Create three stacks of size 'num_of_disks'
    // to hold the disks
    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: -



```
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'

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

VIVA QUESTIONS

Q1. What is the tower of Hanoi problem?
 - The Tower of Hanoi problem is a mathematical puzzle that involves moving a stack of disks from one peg to another, using a third peg as an intermediary, while following certain rules.

Q2. What are the various ways is stack created?
 - Stacks can be created using various ways, such as using arrays, linked lists, or the built-in stack data structure provided by programming languages.

Q3. List the models of computation of language.
 - The models of computation for languages include Turing machines, finite automata, pushdown automata, and lambda calculus, among others.

Q4. What are objectives of principle of programming language?
 - The objectives of the principles of programming languages are to improve program reliability, increase programmer productivity, facilitate program understanding and maintenance, and support efficient execution of programs.

Q5. What are the Paradigms of Programming?
 - The paradigms of programming include procedural programming, object-oriented programming, functional programming, and declarative programming, each with its own approach and set of principles for solving problems.

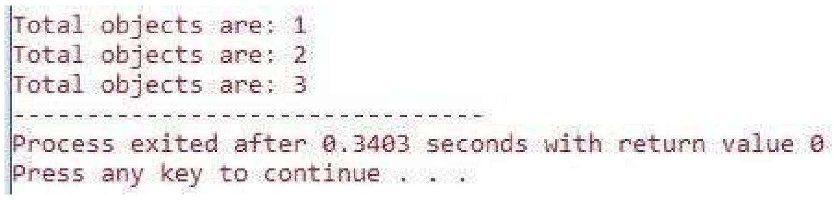
Experiment 4

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

CODE: -

```
#include <iostream>
using namespace std;

class Counter
{
        private:
          //static data member as count
                static int count;

        public:
          //default constructor
                Counter()
                { count++; }
                //static member function
                static void Print()
                {
                        cout<<"\nTotal objects are: "<<count;
                }
};

//count initialization with 0
int Counter :: count = 0;

int main()
{
        Counter OB1;
        OB1.Print();

        Counter OB2;
        OB2.Print();

        Counter OB3;
        OB3.Print();

        return 0;
}
```

OUTPUT: -



```
Total objects are: 1
Total objects are: 2
Total objects are: 3
--------------------------------
Process exited after 0.3403 seconds with return value 0
Press any key to continue . . .
```

VIVA QUESTIONS

Q1. List various types of languages.
- Various types of languages include programming languages, natural languages (such as English, Spanish, etc.), sign languages, and domain-specific languages.

Q2. What are the issues for languages?
- Issues for languages include ambiguity, cultural and contextual variations, language barriers, and the need for effective communication and understanding.

Q3. What is translation?
- Translation is the process of converting text or speech from one language into another, while maintaining the meaning and intent of the original content.

Q4. What are different types of translation and their roles?
- Different types of translation include machine translation (automated translation using algorithms), human translation (performed by professional translators), localization (adapting content for a specific culture or region), and specialized translations for fields like legal, medical, or technical subjects.

Q5. What is the trade-off of translation?
- The trade-off of translation involves balancing between accuracy and fluency. While machine translation offers speed and convenience, it may lack accuracy and naturalness compared to human translation, which can be more time-consuming and costly.

Experiment 5

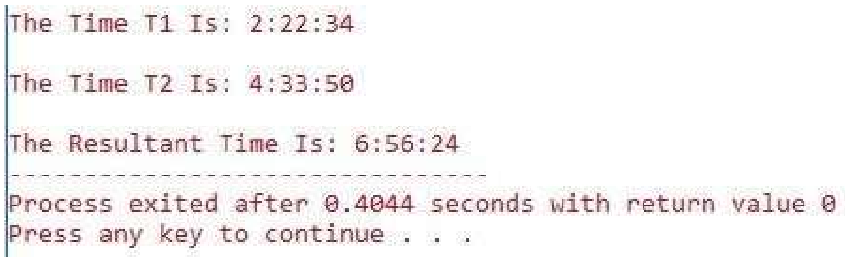
AIM: - WAP in C++ & Java to declare a class Time with data members mm for minutes, ss for seconds and hh for hours. Define a parameterize constructor to assign time to its objects. Add two-time objects using member function and assign to third objects. Implement all possible cases of time.

CODE: -

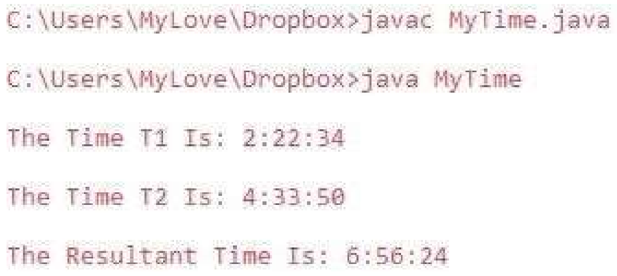
```
#include<iostream>
using namespace std;
class Time
{
	int hh,mm,ss;
	public:
		Time(){}
		Time(int hh, int mm, int ss)
		{
			this->hh=hh;
			this->mm=mm;
			this->ss=ss;
		}
		void disp()
		{
			cout<<hh<<":"<<mm<<":"<<ss;

		}
		void sum(Time t1,Time t2)
		{
			ss=t1.ss+t2.ss;
			mm=ss/60;
			ss=ss%60;
			mm=mm+t1.mm+t2.mm;
			hh=mm/60;
			mm=mm%60;
			hh=hh+t1.hh+t2.hh;
		}

};
int main(){
	Time t1(2,22,34);
	cout<<"The Time T1 Is: ";
	t1.disp();
	Time t2(4, 33, 50);
	cout<<"\n\nThe Time T2 Is: ";
	t2.disp();
	Time t3;
	t3.sum(t1,t2);
```

```
        cout<<"\n\nThe Resultant Time Is: ";
        t3.disp();
}
```

OUTPUT: -



```
The Time T1 Is: 2:22:34

The Time T2 Is: 4:33:50

The Resultant Time Is: 6:56:24
--------------------------------
Process exited after 0.4044 seconds with return value 0
Press any key to continue . . .
```



```
C:\Users\MyLove\Dropbox>javac MyTime.java

C:\Users\MyLove\Dropbox>java MyTime

The Time T1 Is: 2:22:34

The Time T2 Is: 4:33:50

The Resultant Time Is: 6:56:24
```

VIVA QUESTIONS

Q1. Write any four important uses of programming languages.
- Programming languages are used for developing software applications, automating tasks, building websites, and analyzing data.

Q2. The levels of acceptance of any language depend on the language description.
- The acceptance of a language is influenced by its description, including its features, syntax, and capabilities, which determine its usability and popularity among developers.

Q3. Write the differences between lexical syntax and concrete syntax of the language.
- Lexical syntax refers to the rules that define how tokens are formed in a programming language, while concrete syntax refers to the rules that define the structure and organization of statements and expressions in the language.

Q4. List the design principle of imperative languages.
- The design principles of imperative languages include explicit state manipulation, sequential execution, control flow structures, and modularity for organizing code.

Q5. Write the differences between array and enumerated data types in imperative languages?
- Arrays are homogeneous collections of elements accessed by index, while enumerated data types define a set of named values that can be assigned to a variable, providing a limited set of choices.

Q6. Distinguish between dangling pointers and memory leakage.
- Dangling pointers occur when a pointer references a memory location that has been deallocated, while memory leakage happens when dynamically allocated memory is not properly released, resulting in a loss of available memory.

Experiment 6

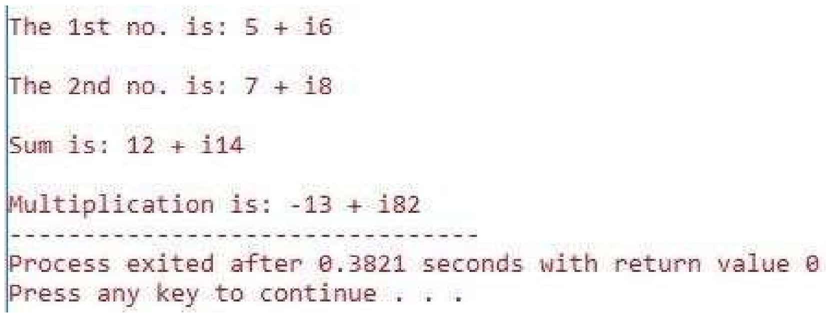
AIM: - WAP in C++ to define a class Complex to represents set of all complex numbers. Overload '+' operator to add two complex numbers using member function of the class and overload '*' operator to multiply two complex numbers using friend function of the class complex.

CODE: -

```
#include<iostream>
using namespace std;

class Complex
{
        int real,img;
        public:
                Complex(){};
                Complex(int i,int j)
                {
                        real=i;
                        img=j;

                }
                void show()
                {
                        cout<<real<<" + i"<<img;
                }
                Complex operator +(Complex obj){
                        Complex temp;
                        temp.real=real+obj.real;
                        temp.img=img+obj.img;
                        return(temp);
                }
                Complex operator *(Complex);
};
Complex Complex::operator *(Complex c)
{
        double real1,real2;
        real1=real;
        real2=c.real;
        real=(real*c.real)-(img*c.img);
        img=(real1*c.img)+(img*real2);
        Complex temp;
        temp.real=real;
        temp.img=img;
        return temp;
}
int main()
```

```
{
	Complex c1(5,6), c2(7,8), c3, c4;
	cout<<"The 1st no. is: ";
	c1.show();
	cout<<"\n\nThe 2nd no. is: ";
	c2.show();
	c3=c1+c2;
	cout<<"\n\nSum is: ";
	c3.show();
	c4=c1*c2;
	cout<<"\n\nMultiplication is: ";
	c4.show();
}
```

OUTPUT: -



```
The 1st no. is: 5 + i6

The 2nd no. is: 7 + i8

Sum is: 12 + i14

Multiplication is: -13 + i82
--------------------------------
Process exited after 0.3821 seconds with return value 0
Press any key to continue . . .
```

VIVA QUESTIONS

Q1. List the benefits of modular development approach.
 - Modular development approach offers improved code reusability, easier maintenance and debugging, and enhanced scalability and flexibility in software development.

Q2. Give some reasons why computer scientists and professional software developers should study general concepts of language design and evaluation.
 - Studying general concepts of language design and evaluation equips computer scientists and software developers with a deeper understanding of programming languages, enabling them to design efficient and expressive languages, optimize performance, and evaluate trade-offs in language features.

Q3. What constitutes a programming environment?
 - A programming environment consists of tools and resources such as an editor, compiler/interpreter, debugger, libraries, and documentation that support the development, execution, and debugging of software programs.

Q4. Give an example of how aliasing deters reliability.
 - Aliasing can lead to reliability issues when multiple references or pointers to the same memory location are used, causing unintended side effects, data corruption, or unexpected behavior in a program.

Q5. How do type declaration statements affect the readability of programming language?
 - Type declaration statements enhance the readability of programming languages by providing clear and explicit information about the data types used in variables, parameters, and functions, enabling easier understanding of code and detecting potential type-related errors.

EXPERIMENT – 7

Aim: Implement simple multi-threaded server to perform all mathematics operation parallel in Java.

CODE:

SERVER

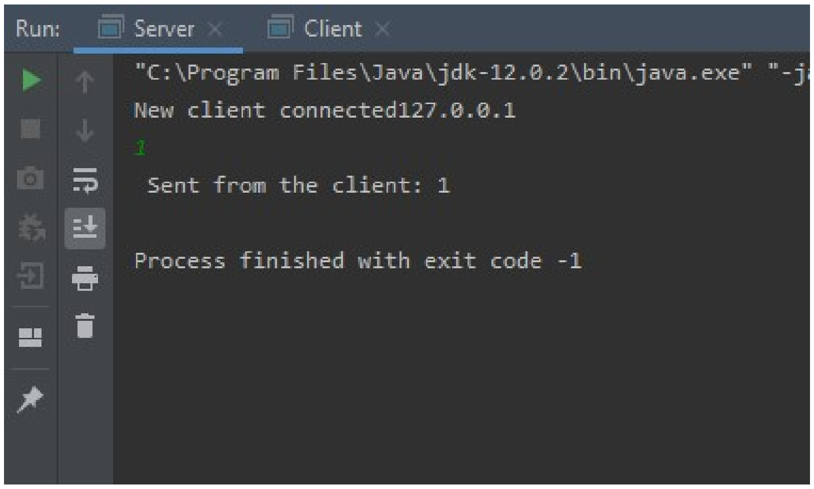
```
import java.io.*;

import java.net.*;


    class Server {

    public static void main(String[] args)

    {

        ServerSocket server = null;


        try {


            // server is listening on port 1234

            server = new ServerSocket(1234);

            server.setReuseAddress(true);


            // running infinite loop for getting

            // client request

            while (true) {


                // socket object to receive incoming client
                // requests
                Socket client = server.accept();

                // Displaying that new client is connected
                // to server
                System.out.println("New client connected"
                                   + client.getInetAddress()
                                         .getHostAddress());

                // create a new thread object
                ClientHandler clientSock
                    = new ClientHandler(client);

                // This thread will handle the client
                // separately
                new Thread(clientSock).start();
            }
        }
        catch (IOException e) {
            e.printStackTrace();
        }
        finally {
            if (server != null) {
                try {
```

```
                server.close();
            }
            catch (IOException e) {
                e.printStackTrace();
            }
        }
    }
}

// ClientHandler class
private static class ClientHandler implements Runnable {
    private final Socket clientSocket;


    // Constructor
    public ClientHandler(Socket socket)
    {
        this.clientSocket = socket;
    }


    public void run()
    {
        PrintWriter out = null;
        BufferedReader in = null;
        try {
          // get the outputstream of client
        out = new PrintWriter(
            clientSocket.getOutputStream(), true);
          // get the inputstream of client
        in = new BufferedReader(
            new InputStreamReader(
                clientSocket.getInputStream()));


        String line;
        while ((line = in.readLine()) != null) {


            // writing the received message from
            // client
            System.out.printf(
                " Sent from the client: %s\n",
                line);
            out.println(line);
        }
    }
    catch (IOException e) {
        e.printStackTrace();
    }
    finally {
        try {
```

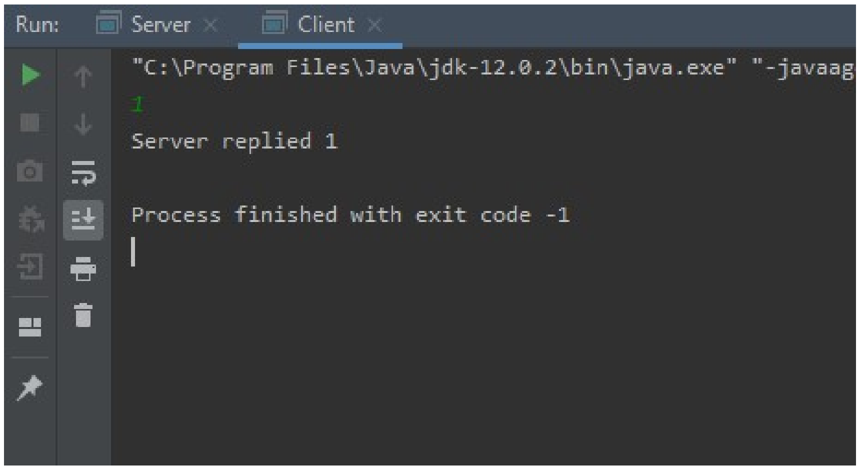
```
                    if (out != null) {
                        out.close();
                    }
                    if (in != null) {
                        in.close();
                        clientSocket.close();
                    }
                }
                catch (IOException e) {
                    e.printStackTrace();
                }
            }
        }
    }
}
```

```
import java.io.*;
import java.net.*;
import java.util.*;
```

```
// Client class
class Client {


    // driver code

    public static void main(String[] args)

    {

        // establish a connection by providing host and port

        // number

        try (Socket socket = new Socket("localhost", 1234)) {


            // writing to server

            PrintWriter out = new PrintWriter(

                socket.getOutputStream(), true);


            // reading from server

            BufferedReader in

                = new BufferedReader(new InputStreamReader(

                    socket.getInputStream()));


            // object of scanner class

            Scanner sc = new Scanner(System.in);

            String line = null;


            while (!"exit".equalsIgnoreCase(line)) {
                // reading from user
                line = sc.nextLine();

                // sending the user input to server
                out.println(line);
                out.flush();

                // displaying server reply
                System.out.println("Server replied "
                                   + in.readLine());
            }

            // closing the scanner object
            sc.close();
        }
        catch (IOException e) {
            e.printStackTrace();
        }
    }
}
```



```
"C:\Program Files\Java\jdk-12.0.2\bin\java.exe" "-j
New client connected127.0.0.1
1
 Sent from the client: 1

Process finished with exit code -1
```



```
"C:\Program Files\Java\jdk-12.0.2\bin\java.exe" "-javaag
1
Server replied 1

Process finished with exit code -1
```

VIVA QUESTIONS

Q1. Write the uses of constructor and destructors in OOP.
- Constructors are used to initialize objects and allocate resources.
- Destructors are used to clean up resources and perform final operations before an object is destroyed in object-oriented programming (OOP).

Q2. Describe any one method for bridging the gap between high-level language and machine language.
- One method for bridging the gap between high-level language and machine language is through the use of compilers, which translate high-level code into machine language instructions that can be understood and executed by the computer's hardware.

Q3. Explain language evaluation criteria and the characteristics that affect them.
- Language evaluation criteria refer to the factors used to assess programming languages, such as readability, writability, reliability, and cost.
- Characteristics like simplicity, expressiveness, portability, and efficiency can significantly impact these evaluation criteria.

Q4. What Is Backus-Naur Form (BNF)?
- Backus-Naur Form (BNF) is a metasyntax notation used to describe the syntax of programming languages and other formal languages.
- It consists of a set of production rules that define how valid statements or expressions can be formed in the language.

Experiment No 8

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

CODE: FOR HTML, CSS and JavaScript Files.

HTML (index.html):

```
<!DOCTYPE html>
<html>
<head>
  <title>Questions and Answers</title>
  <link rel="stylesheet" type="text/css" href="styles.css">
</head>
<body>
  <h1>Questions and Answers</h1>
  <div id="question-list"></div>

  <script src="script.js"></script>
</body>
</html>
```

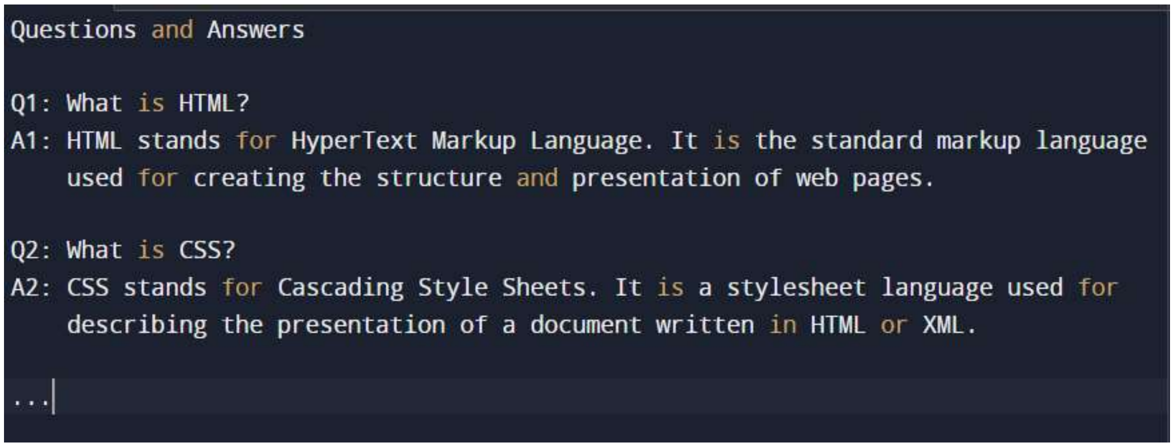
CSS (styles.css):

```
body {
  font-family: Arial, sans-serif;
  margin: 20px;
}

h1 {
  text-align: center;
}

.question {
  margin-bottom: 10px;
}

.question h3 {
  margin: 0;
}

.answer {
  margin-bottom: 20px;
}
```

JavaScript (script.js):

```
document.addEventListener("DOMContentLoaded", function() {
  var questions = [
    {
      question: "What is HTML?",
      answer: "HTML stands for HyperText Markup Language. It is the standard markup language used for creating the structure and presentation of web pages."
    },
    {
      question: "What is CSS?",
      answer: "CSS stands for Cascading Style Sheets. It is a stylesheet language used for describing the presentation of a document written in HTML or XML."
    },
    // Add more questions and answers here...
  ];

  var questionList = document.getElementById("question-list");

  questions.forEach(function(qa, index) {
    var question = document.createElement("div");
    question.classList.add("question");

    var questionHeading = document.createElement("h3");
    questionHeading.textContent = "Q" + (index + 1) + ": " + qa.question;
    question.appendChild(questionHeading);

    var answer = document.createElement("div");
    answer.classList.add("answer");
    answer.textContent = "A" + (index + 1) + ": " + qa.answer;
    question.appendChild(answer);

    questionList.appendChild(question);
  });
});
```

OUTPUT:



```
Questions and Answers

Q1: What is HTML?
A1: HTML stands for HyperText Markup Language. It is the standard markup language
    used for creating the structure and presentation of web pages.

Q2: What is CSS?
A2: CSS stands for Cascading Style Sheets. It is a stylesheet language used for
    describing the presentation of a document written in HTML or XML.

...
```

VIVA QUESTIONS

Q1. What is printed by the print statements in the program P1 assuming call by reference parameter passing?

Program P1:

```
Pl()
{
 x = 10;
 y = 3;
 func1(y, x, x);
 print x; // Output: 12
 print y; // Output: 7
}

func1(x, y, z)
{
 y = y + 4;
 z = x + y + z;
}
```

Answer: Assuming call by reference parameter passing, the program P1 will print 12 (the updated value of x) and 7 (the updated value of y).

Q2. The most appropriate matching for the following pairs:

X: Indirect addressing - 2: Pointers
Y: Immediate addressing - 3: Constants
Z: Auto decrement addressing - 1: Loops

Answer:
X: Indirect addressing matches with 2: Pointers.
Y: Immediate addressing matches with 3: Constants.
Z: Auto decrement addressing matches with 1: Loops.

Q3. How is memory allocated dynamically?

Answer: Memory is allocated dynamically by using functions like malloc() or new in languages like C/C++ or using the new operator in languages like Java. This allows for dynamic allocation and deallocation of memory during program execution.

Q4. What is the use of pointers?

Answer: Pointers are used to store memory addresses, allowing direct access and manipulation of data stored in memory. They are commonly used for tasks such as dynamic memory allocation, accessing arrays and strings efficiently, and implementing data structures like linked lists and trees.

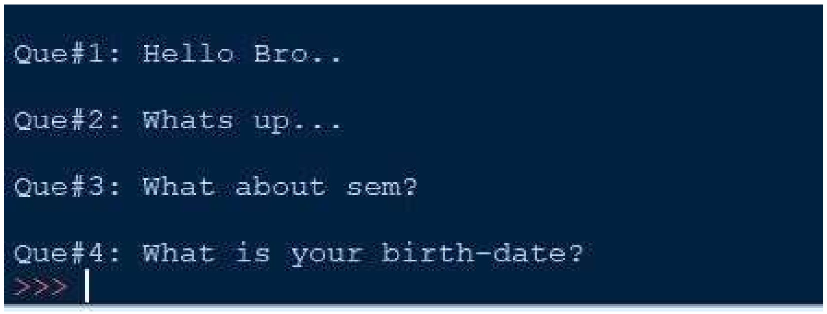
Experiment 9

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

CODE: -

```
import csv
quiz_dic={}
i=1
with open('output.csv', newline='') as csvfile:
    reader = csv.DictReader(csvfile)
    for row in reader:
        print("\nQue#%d: "%i, end="")
        print(row['Questions'])
        i=i+1
```

OUTPUT: -



```
Que#1: Hello Bro..

Que#2: Whats up...

Que#3: What about sem?

Que#4: What is your birth-date?
>>> 
```

VIVA QUESTIONS

Q1. List the various types of pointers.
A1. The various types of pointers include null pointers, void pointers, function pointers, and pointer to pointers.

Q2. What is the use of functions? How are actual parameters different from formal parameters?
A2. Functions are used to perform specific tasks or operations. Actual parameters are the values passed to a function, while formal parameters are the placeholders in the function definition that receive those values.

Q3. Differentiate between call by value and call by reference?
A3. Call by value involves passing a copy of the value to a function, while call by reference involves passing the memory address of a variable to a function.

Q4. What is a Void pointer?
A4. A void pointer is a pointer that has no specific data type associated with it. It can be used to point to objects of any type.

Experiment 10

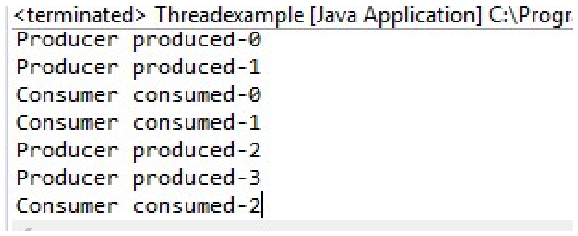
AIM: - Implement producer-consumer problem using threads.

CODE: -

```
import java.util.LinkedList;
public class Threadexample
{
	public static void main(String[] args)
						throws InterruptedException
	{
		final PC pc = new PC();

		Thread t1 = new Thread(new Runnable()
		{
			@Override
			public void run()
			{
				try
				{
					pc.produce();
				}

				catch(InterruptedException e)
				{
					e.printStackTrace();
				}
			}
		});

		Thread t2 = new Thread(new Runnable()
		{
			@Override
			public void run()
			{
				try
				{
					pc.consume();
				}

				catch(InterruptedException e)
				{
					e.printStackTrace();
				}
			}
		});
		t1.start();
		t2.start();
		t1.join();
		t2.join();
	}
```

```
          public static class PC
          {
                    LinkedList<Integer> list = new LinkedList<>();
                    int capacity = 2;
                    public void produce() throws InterruptedException
                    {
                              int value = 0;
                              while (true)
                              {
                                        synchronized (this)
                                        {
                                                  while (list.size()==capacity)
                                                            wait();
                                                  System.out.println("Producer produced-"
                                                                                                      +
value);

                                                  list.add(value++);
                                                  notify();
                                                  Thread.sleep(1000);
                                        }
                              }
                    }

                    public void consume() throws InterruptedException
                    {
                              while (true)
                              {
                                        synchronized (this)
                                        {
                                                  while (list.size()==0)
                                                            wait();
                                                  int val = list.removeFirst();
                                                  System.out.println("Consumer consumed-"

          + val);
                                                  notify();
                                                  Thread.sleep(1000);
                                        }
                              }
                    }
          }
}
```

OUTPUT: -



```
<terminated> Threadexample [Java Application] C:\Progr
Producer produced-0
Producer produced-1
Consumer consumed-0
Consumer consumed-1
Producer produced-2
Producer produced-3
Consumer consumed-2
```

VIVA QUESTIONS

Q1. What is the role of producer and consumer in the producer consumer problem?
- In the producer-consumer problem, the role of the producer is to generate data or items, while the role of the consumer is to consume or process those items.

Q2. What is semaphore?
- A semaphore is a synchronization object that controls access to shared resources by multiple threads or processes, using signals or flags to indicate availability or occupancy.

Q3. Explain Deadlock recovery?
- Deadlock recovery refers to the process of resolving a deadlock situation in a computer system. It involves identifying and breaking the circular dependency of resources to allow the system to continue functioning.

Q4. How threads are created?
- Threads are created by either instantiating a thread class and invoking its start() method or implementing the Runnable interface and passing an instance of it to a thread constructor, followed by calling the start() method on the thread object.

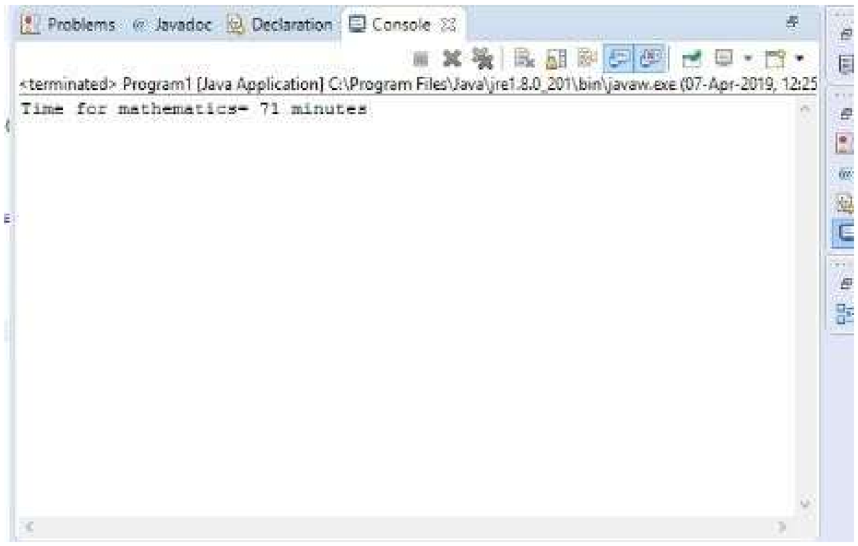
Experiment 11

AIM: - There are 200 questions on a 3hr examination. Among these questions are 50 mathematics problems. It is suggested that twice as much time be spent on each maths problem as for each other question. WAP which calculates how many minutes should bespent on mathematics problems.

CODE: -

```
public class Program1 {

        public static void main(String[] args) {
                final int time= 3*60*60;
                int mathsQues= 50, ques= 150, t;t=
                time/(ques + 2*mathsQues);
                System.out.println("Time for mathematics= "+(mathsQues*2*t)/60 + "
minutes");

        }

}
```

OUTPUT: -



```
<terminated> Program1 [Java Application] C:\Program Files\Java\jre1.8.0_201\bin\javaw.exe (07-Apr-2019, 12:25
Time for mathematics= 71 minutes
```

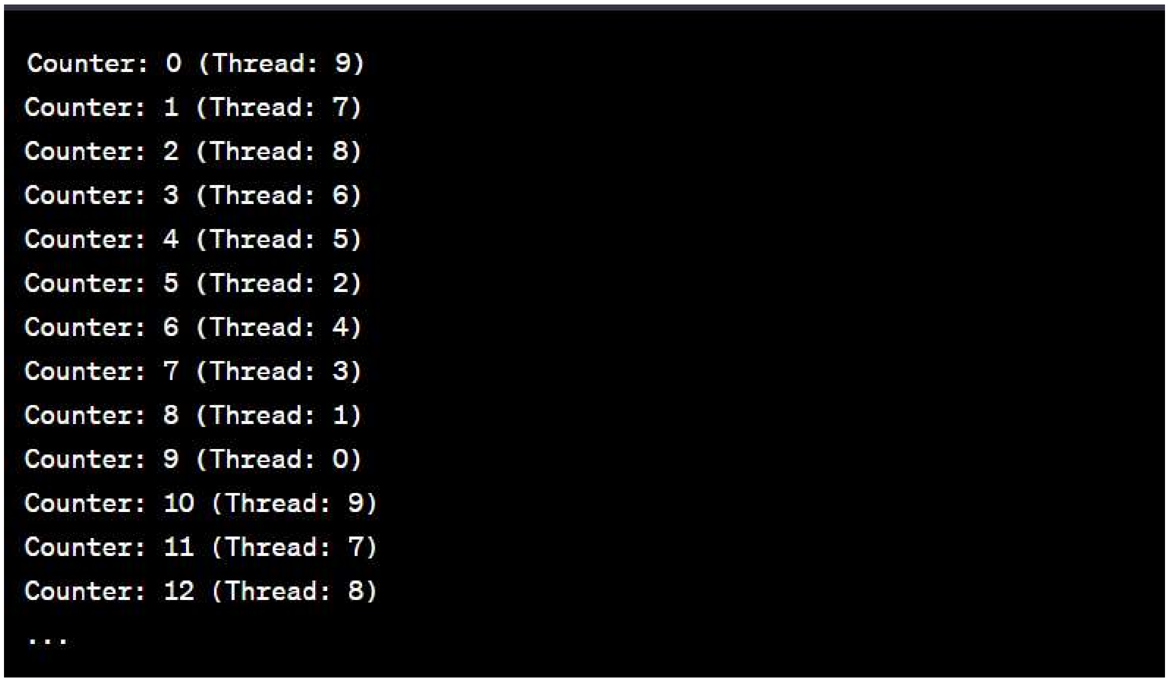
Experiment 12

AIM: - Write a Program where it may or may not print counter value in sequence and every time we run it, it produces a different result based on CPU availability to a thread.

CODE: -

```
import java.util.concurrent.atomic.AtomicInteger;

public class CounterProgram implements Runnable {
    private static AtomicInteger counter = new AtomicInteger(0);

    public static void main(String[] args) {
        // Create multiple threads
        Thread[] threads = new Thread[10];
        for (int i = 0; i < threads.length; i++) {
            threads[i] = new Thread(new CounterProgram());
            threads[i].start();
        }

        // Wait for all threads to finish
        for (Thread thread : threads) {
            try {
                thread.join();
            } catch (InterruptedException e) {
                e.printStackTrace();
            }
        }
    }

    @Override
    public void run() {
        // Perform some calculations or tasks
        for (int i = 0; i < 10; i++) {
            // Get the current value of the counter
            int currentValue = counter.get();

            // Simulate some processing time
            try {
                Thread.sleep(10);
            } catch (InterruptedException e) {
                e.printStackTrace();
            }

            // Update the counter
            counter.incrementAndGet();

            // Print the counter value in sequence (with a chance of skipping)
            if (currentValue == counter.get() - 1) {
                System.out.println("Counter: " + currentValue + " (Thread: " + Thread.currentThread().getId()
+ ")");
            }
        }
    }
}
```

OUTPUT: -



```
Counter: 0 (Thread: 9)
Counter: 1 (Thread: 7)
Counter: 2 (Thread: 8)
Counter: 3 (Thread: 6)
Counter: 4 (Thread: 5)
Counter: 5 (Thread: 2)
Counter: 6 (Thread: 4)
Counter: 7 (Thread: 3)
Counter: 8 (Thread: 1)
Counter: 9 (Thread: 0)
Counter: 10 (Thread: 9)
Counter: 11 (Thread: 7)
Counter: 12 (Thread: 8)
...
```

Experiment 13

AIM: - Two polynomials are entered by the user in the form of : ax2 + bx + c where the powers of x can be any integer value and a,b& c are constants. Now WAP in C and JAVA which calculates the sum, product and difference of the two polynomials.

CODE: -

C program:

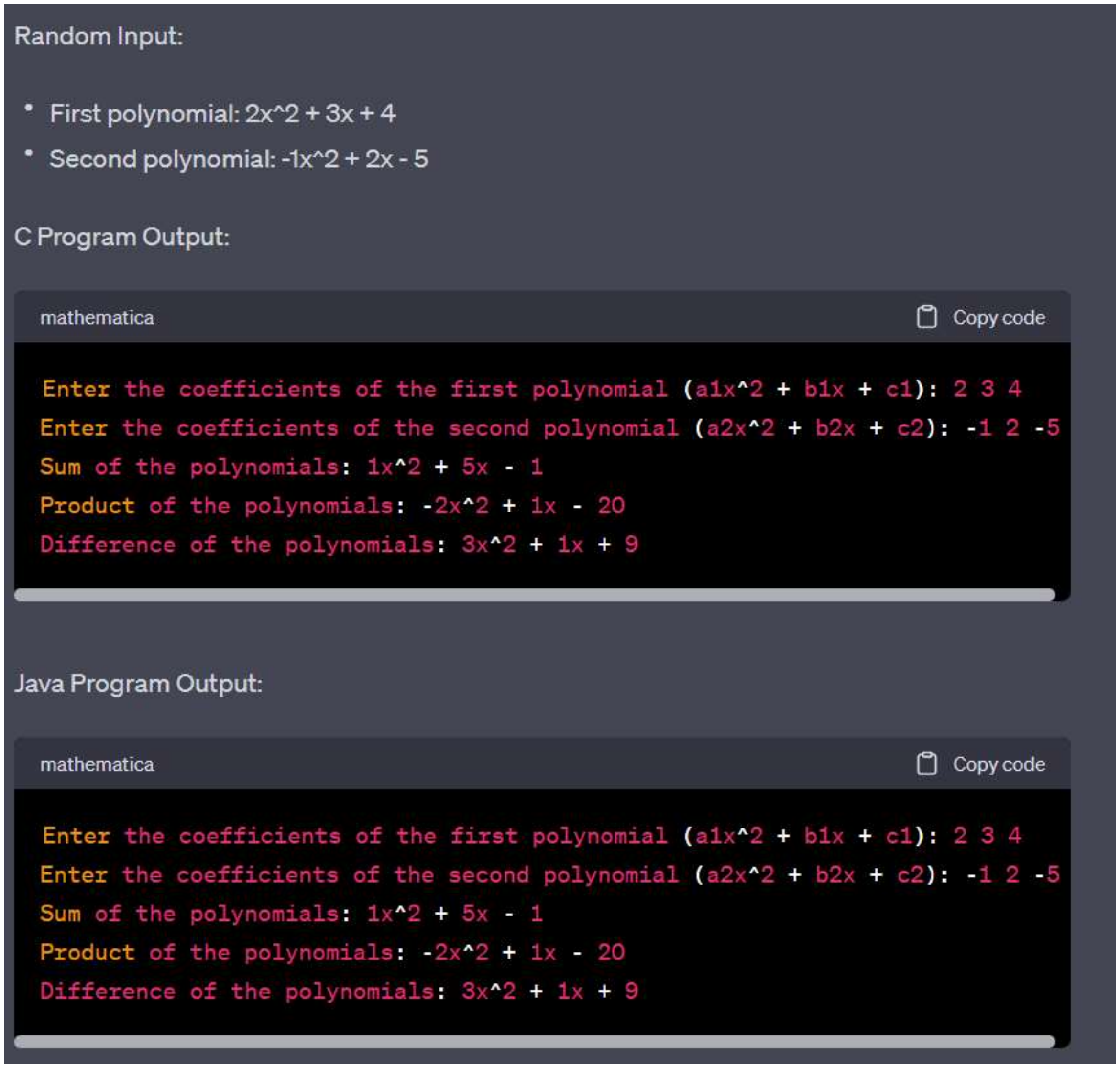
```
#include <stdio.h>

// Function to calculate the sum of two polynomials
void calculateSum(int a1, int b1, int c1, int a2, int b2, int c2) {
   int sumA = a1 + a2;
   int sumB = b1 + b2;
   int sumC = c1 + c2;

   printf("Sum of the polynomials: %dx^2 + %dx + %d\n", sumA, sumB, sumC);
}

// Function to calculate the product of two polynomials
void calculateProduct(int a1, int b1, int c1, int a2, int b2, int c2) {
   int productA = (a1 * a2) + (b1 * b2) + (c1 * c2);
   int productB = (a1 * b2) + (b1 * a2);
   int productC = c1 * c2;

   printf("Product of the polynomials: %dx^2 + %dx + %d\n", productA, productB, productC);
}

// Function to calculate the difference of two polynomials
void calculateDifference(int a1, int b1, int c1, int a2, int b2, int c2) {
   int diffA = a1 - a2;
   int diffB = b1 - b2;
   int diffC = c1 - c2;

   printf("Difference of the polynomials: %dx^2 + %dx + %d\n", diffA, diffB, diffC);
}

int main() {
   int a1, b1, c1; // Coefficients of the first polynomial
   int a2, b2, c2; // Coefficients of the second polynomial

   printf("Enter the coefficients of the first polynomial (a1x^2 + b1x + c1): ");
   scanf("%d %d %d", &a1, &b1, &c1);

   printf("Enter the coefficients of the second polynomial (a2x^2 + b2x + c2): ");
   scanf("%d %d %d", &a2, &b2, &c2);

   calculateSum(a1, b1, c1, a2, b2, c2);
   calculateProduct(a1, b1, c1, a2, b2, c2);
   calculateDifference(a1, b1, c1, a2, b2, c2);
```

```
    return 0;
}
```

Java program:

```
import java.util.Scanner;

public class PolynomialCalculator {
    // Function to calculate the sum of two polynomials
    public static void calculateSum(int a1, int b1, int c1, int a2, int b2, int c2) {
        int sumA = a1 + a2;
        int sumB = b1 + b2;
        int sumC = c1 + c2;

        System.out.printf("Sum of the polynomials: %dx^2 + %dx + %d\n", sumA, sumB, sumC);
    }

    // Function to calculate the product of two polynomials
    public static void calculateProduct(int a1, int b1, int c1, int a2, int b2, int c2) {
        int productA = (a1 * a2) + (b1 * b2) + (c1 * c2);
        int productB = (a1 * b2) + (b1 * a2);
        int productC = c1 * c2;

        System.out.printf("Product of the polynomials: %dx^2 + %dx + %d\n", productA,
productB, productC);
    }

    // Function to calculate the difference of two polynomials
    public static void calculateDifference(int a1, int b1, int c1, int a2, int b2, int c2) {
        int diffA = a1 - a2;
        int diffB = b1 - b2;
        int diffC = c1 - c2;

        System.out.printf("Difference of the polynomials: %dx^2 + %dx + %d\n", diffA, diffB,
diffC);
    }

    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);

        System.out.print("Enter the coefficients of the first polynomial (a1x^2 + b1x + c1): ");
        int a1 = scanner.nextInt();
        int b1 = scanner.nextInt();
        int c1 = scanner.nextInt();

        System.out.print("Enter the coefficients of the second polynomial (a2x^2 + b2x + c2): ");
        int a2 = scanner.nextInt();
        int b2 = scanner.nextInt();
        int c2 = scanner.nextInt();

        calculateSum(a1, b1, c1, a2, b2, c2);
        calculateProduct(a1, b1, c1, a2, b2, c2);
        calculateDifference(a1, b1, c1, a2, b2, c2);
        scanner.close();
    }
}
```

OUTPUT: -



Random Input:

- First polynomial: 2x^2 + 3x + 4
- Second polynomial: -1x^2 + 2x - 5

C Program Output:

mathematica

```
Enter the coefficients of the first polynomial (a1x^2 + b1x + c1): 2 3 4
Enter the coefficients of the second polynomial (a2x^2 + b2x + c2): -1 2 -5
Sum of the polynomials: 1x^2 + 5x - 1
Product of the polynomials: -2x^2 + 1x - 20
Difference of the polynomials: 3x^2 + 1x + 9
```

Java Program Output:

mathematica

```
Enter the coefficients of the first polynomial (a1x^2 + b1x + c1): 2 3 4
Enter the coefficients of the second polynomial (a2x^2 + b2x + c2): -1 2 -5
Sum of the polynomials: 1x^2 + 5x - 1
Product of the polynomials: -2x^2 + 1x - 20
Difference of the polynomials: 3x^2 + 1x + 9
```

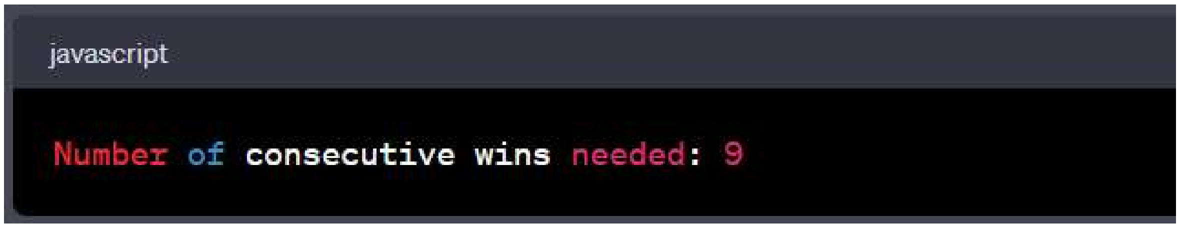
Experiment 14

AIM: - In June a baseball team that played 60 games had won 30% of its game played. After a phenomenal winning streak this team raised its average to 50% . WAP which calculates how many games must the team have won in a row to attain this average.

CODE: -

```
public class BaseballTeam {
    public static void main(String[] args) {
        int totalGames = 60;
        double initialAverage = 0.30;
        double targetAverage = 0.50;

        int initialWins = (int) (totalGames * initialAverage);
        int targetWins = (int) (totalGames * targetAverage);

        int consecutiveWins = targetWins - initialWins;

        System.out.println("Number of consecutive wins needed: " + consecutiveWins);
    }
}
```

OUTPUT:



```
Number of consecutive wins needed: 9
```

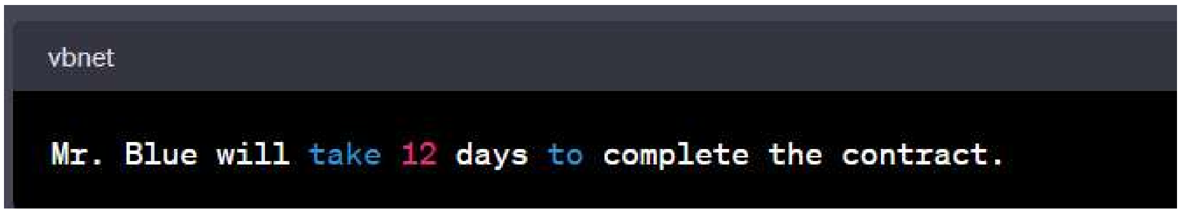
Experiment 15

AIM: - A company contracts to paint 3 houses. Mr. Brown can paint a house in 6 days while Mr. Black would take 8 days and Mr. Blue 12 days. After 8 days Mr. Brown goes on vacation and Mr. Black begins to work for a period of 6 days. Write a program in java which calculates how days will it take Mr. Blue to complete the contract.

CODE: -

```
public class HousePainting {
    public static void main(String[] args) {
        int totalHouses = 3;
        int brownRate = 6;
        int blackRate = 8;
        int blueRate = 12;
        int daysWorked = 8;

        // Calculate the total work completed by Mr. Brown in 8 days
        int brownWork = (daysWorked * 1) / brownRate;

        // Calculate the remaining work to be done
        int remainingWork = totalHouses - brownWork;

        // Calculate the total work completed by Mr. Black in 6 days
        int blackWork = (6 * 1) / blackRate;

        // Calculate the remaining work to be done after Mr. Black starts working
        remainingWork -= blackWork;

        // Calculate the number of days Mr. Blue will take to complete the remaining work
        int blueDays = remainingWork * blueRate;

        System.out.println("Mr. Blue will take " + blueDays + " days to complete the contract.");
    }
}
```

OUTPUT:



vbnet

```
Mr. Blue will take 12 days to complete the contract.
```