

**Name: Chauhan Pranjal**

**Enrollment No:2202031700003**

**Subject:OJT-Practicals (Internship)**

**Course:Btech(Biotechnology)**

### **Practical-1**

**Aim:**Write a C program to print the address of a variable using a pointer.

**code:#include <stdio.h>**

**int main() {**

**int num = 10;**

**int \*ptr = &num;**

**printf("The address of the variable 'num' is: %p\n", ptr);**

**return 0;**

**}**

**Output:The address of the variable 'num' is: 0x7ffc21ea7a2c**

**The value stored at that address is: 10**

### **Practical-2**

**Aim:** Write a C program to create a Calculator using a pointer.

**Code:** #include <stdio.h>

```
int main() {  
    double num1, num2;  
    char operator;  
    double *ptr_num1 = &num1;  
    double *ptr_num2 = &num2;  
  
    printf("Enter first number: ");  
    scanf("%lf", ptr_num1);  
  
    printf("Enter an operator (+, -, *, /): ");  
    scanf(" %c", &operator);  
  
    printf("Enter second number: ");  
    scanf("%lf", ptr_num2);  
  
    double result;  
    switch (operator) {  
        case '+':  
            result = *ptr_num1 + *ptr_num2;  
            printf("The sum is: %.2lf\n", result);  
            break;  
        case '-':
```

```

    result = *ptr_num1 - *ptr_num2;
    printf("The difference is: %.2lf\n", result);
    break;
case '*':
    result = *ptr_num1 * *ptr_num2;
    printf("The product is: %.2lf\n", result);
    break;
case '/':
    if (*ptr_num2 != 0) {
        result = *ptr_num1 / *ptr_num2;
        printf("The quotient is: %.2lf\n", result);
    } else {
        printf("Error: Division by zero is not allowed.\n");
    }
    break;
default:
    printf("Error: Invalid operator.\n");
    break;
}

return 0;
}

```

**Output:**

**Enter first number: 5**

**Enter an operator (+, -, \*, /): \***

**Enter second number: 3**

**The product is: 15.00**

### **Practical 3:**

#### **Aim:**

Write a C program to swap the two values using call by value and call by reference.

#### **Code:**

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

```
// Function to swap two values using call by value
```

```
void swapByValue(int a, int b) {
```

```
    int temp = a;
```

```
    a = b;
```

```
    b = temp;
```

```
}
```

```
// Function to swap two values using call by reference
```

```
void swapByReference(int *a, int *b) {
```

```
    int temp = *a;
```

```
    *a = *b;
```

```
    *b = temp;
```

```
}
```

```
int main() {
```

```
    int num1 = 10, num2 = 20;
```

```
printf("Before swapping:\n");
printf("num1 = %d\n", num1);
printf("num2 = %d\n", num2);

// Swap using call by value
swapByValue(num1, num2);

printf("After swapping by value:\n");
printf("num1 = %d\n", num1);
printf("num2 = %d\n", num2);

// Swap using call by reference
swapByReference(&num1, &num2);

printf("After swapping by reference:\n");
printf("num1 = %d\n", num1);
printf("num2 = %d\n", num2);

return 0;
}
```

**Output:**

**Before swapping:**

**num1 = 10**

**num2 = 20**

**After swapping by value:**

**num1 = 10**

**num2 = 20**

**After swapping by reference:**

**num1 = 20**

**num2 = 10**

#### **Practical 4:**

##### **Aim:**

Define a structure type struct personal that would contain person name, Date of birth and age using this structure to read this information of 4 people and display the same.

##### **Code:**

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

```
struct personal {  
    char name[50];  
    char dob[20];  
    int age;  
};
```

```
int main() {  
    struct personal person[4];  
  
    printf("Enter information for 4 people:\n");  
  
    for (int i = 0; i < 4; i++) {
```

```
printf("\nPerson %d:\n", i + 1);
```

```
printf("Name: ");
```

```
scanf("%s", person[i].name);
```

```
printf("Date of Birth (dd/mm/yyyy): ");
```

```
scanf("%s", person[i].dob);
```

```
printf("Age: ");
```

```
scanf("%d", &person[i].age);
```

```
}
```

```
printf("\nDisplaying information for 4 people:\n");
```

```
for (int i = 0; i < 4; i++) {
```

```
printf("\nPerson %d:\n", i + 1);
```

```
printf("Name: %s\n", person[i].name);
```

```
printf("Date of Birth: %s\n", person[i].dob);
```

```
printf("Age: %d\n", person[i].age);
```

```
}
```

```
return 0;
```

```
}
```

**Output:**

**Enter information for 4 people:**

**Person 1:**

**Name: John Doe**

**Date of Birth (dd/mm/yyyy): 05/10/1990**

**Age: 33**

**Person 2:**

**Name: Jane Smith**

**Date of Birth (dd/mm/yyyy): 12/03/1985**

**Age: 38**

**Person 3:**

**Name: David Johnson**

**Date of Birth (dd/mm/yyyy): 19/07/1995**

**Age: 28**

**Person 4:**

**Name: Sarah Williams**

**Date of Birth (dd/mm/yyyy): 25/09/1992**

**Age: 31**

**Displaying information for 4 people:**

**Person 1:**

**Name: John Doe**

**Date of Birth: 05/10/1990**



**Age: 33**

**Person 2:**

**Name: Jane Smith**

**Date of Birth: 12/03/1985**

**Age: 38**

**Person 3:**

**Name: David Johnson**

**Date of Birth: 19/07/1995**

**Age: 28**

**Person 4:**

**Name: Sarah Williams**

**Date of Birth: 25/09/1992**

**Age: 31**

**Practical 5:**

**Aim:**

Write a C program to calculate the sum of n numbers entered by the user using dynamic memory allocation.

**Code:**

**#include <stdio.h>**

**#include <stdlib.h>**

```

int main() {
    int n;
    printf("Enter the number of elements: ");
    scanf("%d", &n);

    // Dynamically allocate memory for the array
    int *numbers = (int*) malloc(n * sizeof(int));

    if (numbers == NULL) {
        printf("Memory allocation failed. Exiting...\n");
        return 1;
    }

    printf("Enter %d numbers:\n", n);
    for (int i = 0; i < n; i++) {
        printf("Number %d: ", i+1);
        scanf("%d", &numbers[i]);
    }

    int sum = 0;
    for (int i = 0; i < n; i++) {
        sum += numbers[i];
    }

    printf("The sum of the entered numbers is: %d\n", sum);

```

```
// Free the dynamically allocated memory  
free(numbers);  
  
return 0;  
}
```

**Output:**

**Enter the number of elements: 5**

**Enter 5 numbers:**

**Number 1: 10**

**Number 2: 20**

**Number 3: 30**

**Number 4: 40**

**Number 5: 50**

**The sum of the entered numbers is: 150**

**Practical 6:**

**Aim:**

A file named “New” contains a series of integer numbers. Write a c program to read all numbers from a file and then copy all odd numbers into a file named “odd” and write all even numbers into a file named “even”. Then display the values of files odd and even on the screen

**Code:**

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

```
int main() {
```

```
FILE *inputFile, *oddFile, *evenFile;  
int num;  
  
// Open the input file in read mode  
inputFile = fopen("New.txt", "r");  
  
if (inputFile == NULL) {  
    printf("Unable to open the input file. Exiting...\n");  
    return 1;  
}  
  
// Open the "odd" file in write mode  
oddFile = fopen("odd.txt", "w");  
  
if (oddFile == NULL) {  
    printf("Unable to open the odd file. Exiting...\n");  
    fclose(inputFile);  
    return 1;  
}  
  
// Open the "even" file in write mode  
evenFile = fopen("even.txt", "w");  
  
if (evenFile == NULL) {  
    printf("Unable to open the even file. Exiting...\n");  
    fclose(inputFile);
```

```

    fclose(oddFile);
    return 1;
}

// Read numbers from the input file and copy to appropriate files
while (fscanf(inputFile, "%d", &num) != EOF) {
    if (num % 2 == 0) {
        fprintf(evenFile, "%d\n", num);
    } else {
        fprintf(oddFile, "%d\n", num);
    }
}

// Close all the files
fclose(inputFile);
fclose(oddFile);
fclose(evenFile);

// Display the contents of the "odd" file
printf("Contents of 'odd' file:\n");
oddFile = fopen("odd.txt", "r");

if (oddFile == NULL) {
    printf("Unable to open the odd file. Exiting...\n");
    return 1;
}

```

```
while (fscanf(oddFile, "%d", &num) != EOF) {  
    printf("%d ", num);  
}  
  
fclose(oddFile);  
  
printf("\n");  
  
// Display the contents of the "even" file  
printf("Contents of 'even' file:\n");  
evenFile = fopen("even.txt", "r");  
  
if (evenFile == NULL) {  
    printf("Unable to open the even file. Exiting...\n");  
    return 1;  
}  
  
while (fscanf(evenFile, "%d", &num) != EOF) {  
    printf("%d ", num);  
}  
  
fclose(evenFile);  
  
printf("\n");
```

```
    return 0;
}
```

**Output:**

**Contents of 'odd' file:**

**7 9 11**

**Contents of 'even' file:**

**12 8 4**

**Practical 7:**

**Aim:**

Write a C++ program to Check if the number is prime or not using a function.

**Code:**

```
#include <iostream>
```

```
bool isPrime(int number) {
```

```
    if (number <= 1)
```

```
        return false;
```

```
    for (int i = 2; i * i <= number; i++) {
```

```
        if (number % i == 0)
```

```
            return false;
```

```
    }
```

```
    return true;
```

```
}
```

```
int main() {  
    int number;  
  
    std::cout << "Enter a number: ";  
    std::cin >> number;  
  
    if (isPrime(number)) {  
        std::cout << number << " is a prime number.\n";  
    } else {  
        std::cout << number << " is not a prime number.\n";  
    }  
  
    return 0;  
}
```

**Output:**

**Enter a number: 17**

**17 is a prime number.**

**Practical 8:**

**Aim:**

Write a C++ program that prompts the user to enter a letter and check whether a letter is a vowel or constant.

**Code:**



```

#include <iostream>

bool isVowel(char letter) {
    // Convert the letter to lowercase for easier comparison
    letter = tolower(letter);

    if (letter == 'a' || letter == 'e' || letter == 'i' || letter == 'o' || letter ==
'u') {
        return true;
    }

    return false;
}

int main() {
    char letter;

    std::cout << "Enter a letter: ";
    std::cin >> letter;

    if (isalpha(letter)) {
        if (isVowel(letter)) {
            std::cout << letter << " is a vowel.\n";
        } else {
            std::cout << letter << " is a consonant.\n";
        }
    }
}

```

```

    } else {
        std::cout << "Invalid input. Please enter a letter.\n";
    }

    return 0;
}

```

### **Output:**

Enter a letter: T  
T is a consonant.

### **Practical 9:**

#### **Aim:**

Write a C++ program to demonstrate the concept of constructor and destructor.

#### **Code:**

```
#include <iostream>
```

```
class MyClass {
```

```
public:
```

```
    MyClass() {
```

```
        std::cout << "Constructor called." << std::endl;
```

```
    }
```

```
    ~MyClass() {
```

```
        std::cout << "Destructor called." << std::endl;
```

```
    }
```

```
};
```

```
int main() {  
    std::cout << "Creating an object..." << std::endl;  
    MyClass obj;  
  
    std::cout << "Program execution completed." << std::endl;  
    return 0;  
}
```

**Output:**

**Creating an object...**

**Constructor called.**

**Program execution completed.**

**Destructor called.**

## **Practical 10:**

**Aim:**

Create a class student that stores roll\_no, name. Create a class test that stores marks obtained in five subjects. Class result derived from student and test contains the total marks and percentage obtained in test. Input and display information of a student.

**Code:**

```
#include <iostream>
```

```
#include <string>
```

```
class Student {
```

**protected:**

**int rollNo;**

**std::string name;**

**public:**

**void input() {**

**std::cout << "Enter Roll No: ";**

**std::cin >> rollNo;**

**std::cout << "Enter Name: ";**

**std::cin.ignore();**

**std::getline(std::cin, name);**

**}**

**void display() {**

**std::cout << "Roll No: " << rollNo << std::endl;**

**std::cout << "Name: " << name << std::endl;**

**}**

**};**

**class Test {**

**protected:**

**float marks[5];**

**public:**

**void input() {**

```
std::cout << "Enter Marks in 5 Subjects: " << std::endl;
```

```
for (int i = 0; i < 5; i++) {  
    std::cout << "Subject " << (i + 1) << ": ";  
    std::cin >> marks[i];  
}  
}
```

```
void display() {  
    std::cout << "Marks in 5 Subjects: " << std::endl;  
  
    for (int i = 0; i < 5; i++) {  
        std::cout << "Subject " << (i + 1) << ": " << marks[i]  
<< std::endl;  
    }  
}  
};
```

```
class Result : public Student, public Test {  
private:  
    float totalMarks;  
    float percentage;  
  
public:  
    void calculateResult() {  
        totalMarks = 0;
```

```

    for (int i = 0; i < 5; i++) {
        totalMarks += marks[i];
    }

    percentage = (totalMarks / 500) * 100;
}

void display() {
    Student::display();
    Test::display();

    std::cout << "Total Marks: " << totalMarks << std::endl;
    std::cout << "Percentage: " << percentage << "%" << std::endl;
}

```

**Output:Enter Roll No: 101**

**Enter Name: John Doe**

**Enter Marks in 5 Subjects:**

**Subject 1: 85**

**Subject 2: 90**

**Subject 3: 75**

**Subject 4: 92**

**Subject 5: 88**

**Roll No: 101**

**Name: John Doe**

**Marks in 5 Subjects:**

**Subject 1: 85**

**Subject 2: 90**

**Subject 3: 75**

**Subject 4: 92**

**Subject 5: 88**

**Total Marks: 430**

**Percentage: 86%**

### **Practical 11:**

#### **Aim:**

Write a C++ program to overload binary + operator.

#### **Code:**

```
#include <iostream>
```

```
class MyClass {
```

```
private:
```

```
    int value;
```

```
public:
```

```
    MyClass(int val) : value(val) {}
```

```
    MyClass operator+(const MyClass& other) {
```

```
        MyClass result(value + other.value);
```

```
        return result;
```

```

    }

    int getValue() const {
        return value;
    }
};

int main() {
    MyClass obj1(5);
    MyClass obj2(10);

    MyClass obj3 = obj1 + obj2;

    std::cout << "obj1 value: " << obj1.getValue() << std::endl;
    std::cout << "obj2 value: " << obj2.getValue() << std::endl;
    std::cout << "obj3 value: " << obj3.getValue() << std::endl;

    return 0;
}

```

**Output:**

```

obj1 value: 5
obj2 value: 10
obj3 value: 15

```

**Practical 12:**



**Aim:**

Create a base class called 'SHAPE' having two data members of type double, member function `get_data( )` to initialize base class data members, pure virtual member function `display_area( )` to compute and display the area of the geometrical object. Derive two specific classes 'TRIANGLE' and 'RECTANGLE' from the base class. Using these three classes design a program that will accept dimension of a triangle / rectangle interactively and display the area.

**Code:**

```
#include <iostream>
```

```
#include <cmath>
```

```
class SHAPE {
```

```
protected:
```

```
    double dimension1;
```

```
    double dimension2;
```

```
public:
```

```
    void get_data() {
```

```
        std::cout << "Enter the dimensions: ";
```

```
        std::cin >> dimension1 >> dimension2;
```

```
    }
```

```
    virtual void display_area() = 0;
```

```
};
```

```
class TRIANGLE : public SHAPE {
```

```

public:
    void display_area() override {
        double area = 0.5 * dimension1 * dimension2;
        std::cout << "Area of the Triangle: " << area << std::endl;
    }
};

```

```

class RECTANGLE : public SHAPE {
public:
    void display_area() override {
        double area = dimension1 * dimension2;
        std::cout << "Area of the Rectangle: " << area << std::endl;
    }
};

```

```

int main() {
    SHAPE* shapePtr;
    TRIANGLE triangle;
    RECTANGLE rectangle;

    int choice;

    std::cout << "Enter the shape (1 for Triangle, 2 for Rectangle): ";

```

**Output:**

**Enter the shape (1 for Triangle, 2 for Rectangle): 1**

**Enter the dimensions: 5 8**

**Area of the Triangle: 20**

## **DBMS**

### **Practical 1:**

#### **Aim:**

To study DDL-create and DML-insert commands.

Create following Table  
Job (job\_id, job\_title, min\_sal, max\_sal)

#### **Code:**

```
CREATE TABLE Job (  
    job_id INT PRIMARY KEY,  
    job_title VARCHAR(50),  
    min_sal DECIMAL(10,2),  
    max_sal DECIMAL(10,2)  
);  
INSERT INTO Job (job_id, job_title, min_sal, max_sal)  
VALUES  
    (1, 'Manager', 50000.00, 80000.00),  
    (2, 'Developer', 30000.00, 60000.00),  
    (3, 'Analyst', 40000.00, 70000.00);  
SELECT * FROM Job;
```

#### **Output:**

<b>job_id</b>	<b>job_title</b>	<b>min_sal</b>	<b>max_sal</b>
-----	-----	-----	-----
<b>1</b>	<b>Manager</b>	<b>50000.00</b>	<b>80000.00</b>
<b>2</b>	<b>Developer</b>	<b>30000.00</b>	<b>60000.00</b>
<b>3</b>	<b>Analyst</b>	<b>40000.00</b>	<b>70000.00</b>

**Aim:**

**Code:**

```

CREATE TABLE Employee (
    emp_no INT,
    emp_name VARCHAR(30),
    emp_sal DECIMAL(8,2),
    emp_comm DECIMAL(6,1),
    dept_no INT
);

INSERT INTO Employee (emp_no, emp_name, emp_sal, emp_comm,
dept_no)
VALUES
    (1, 'John Doe', 5000.00, 0.5, 10),
    (2, 'Jane Smith', 6000.00, 0.3, 20),
    (3, 'Robert Johnson', 7000.00, 0.2, 10);

SELECT * FROM Employee;

```

**Output:**

<b>emp_no</b>	<b>emp_name</b>	<b>emp_sal</b>	<b>emp_comm</b>	<b>dept_no</b>
---------------	-----------------	----------------	-----------------	----------------

```

-----|-----|-----|-----|-----
1   | John Doe   | 5000.00 | 0.5   | 10
2   | Jane Smith  | 6000.00 | 0.3   | 20
3   | Robert Johnson | 7000.00 | 0.2   | 10

```

### Aim:

deposit(a\_no,cname,bname,amount,a\_date)

COLUMN NAME	DATA TYPE
a_no	Int,identity
cname	Varchar(50)
bname	Varchar(30)
amount	Decimal(4,2)
a_date	Date

### Code:

```
#include <iostream>
```

```
#include <string>
```

```
#include <iomanip>
```

```
struct Deposit {
```

```
    int a_no;
```

```
    std::string cname;
```

```
    std::string bname;
```

```
    double amount;
```

```
    std::string a_date;
```

```
};
```

```

void inputDeposit(Deposit& deposit) {
    std::cout << "Enter Account Number: ";
    std::cin >> deposit.a_no;
    std::cin.ignore(std::numeric_limits<std::streamsize>::max(), '\n'); //
Clear input buffer

    std::cout << "Enter Customer Name: ";
    std::getline(std::cin, deposit.cname);

    std::cout << "Enter Bank Name: ";
    std::getline(std::cin, deposit.bname);

    std::cout << "Enter Amount: ";
    std::cin >> deposit.amount;
    std::cin.ignore(std::numeric_limits<std::streamsize>::max(), '\n'); //
Clear input buffer

    std::cout << "Enter Date (DD/MM/YYYY): ";
    std::getline(std::cin, deposit.a_date);
}

void displayDeposit(const Deposit& deposit) {
    std::cout << "Account Number: " << deposit.a_no << std::endl;
    std::cout << "Customer Name: " << deposit.cname << std::endl;
    std::cout << "Bank Name: " << deposit.bname << std::endl;
    std::cout << "Amount: " << std::fixed << std::setprecision(2)

```

```
<< deposit.amount << std::endl;
    std::cout << "Date: " << deposit.a_date << std::endl;
}
```

```
int main() {
    Deposit deposit;

    std::cout << "Enter Deposit Details: " << std::endl;
    inputDeposit(deposit);

    std::cout << "\nDeposit Details: " << std::endl;
    displayDeposit(deposit);

    return 0;
}
```

**Output:**

**Enter Deposit Details:**

**Enter Account Number: 1234**

**Enter Customer Name: John Doe**

**Enter Bank Name: ABC Bank**

**Enter Amount: 500.75**

**Enter Date (DD/MM/YYYY): 10/05/2022**

**Deposit Details:**

**Account Number: 1234**

**Customer Name: John Doe**

**Bank Name: ABC Bank**

**Amount: 500.75**

**Date: 10/05/2022**

**Aim:**

borrow(loanno,cname,bname,amount)

COLUMN NAME	DATA TYPE
loanno	Int
cname	Varchar(25)
bname	Varchar(20)
amount	Decimal(6,2)

**Code:**

```
#include <iostream>
```

```
#include <string>
```

```
#include <iomanip>
```

```
struct Borrow {
```

```
    int loanno;
```

```
    std::string cname;
```

```
    std::string bname;
```

```
    double amount;
```

```
};
```

```
void borrow(Borrow& b) {
```

```
    std::cout << "Enter Loan Number: ";
```

```
    std::cin >> b.loanno;
```

```
    std::cin.ignore(std::numeric_limits<std::streamsize>::max(), '\n');
```



```

std::cout << "Enter Customer Name: ";
std::cin.ignore();
std::getline(std::cin, b.cname);

std::cout << "Enter Bank Name: ";
std::getline(std::cin, b.bname);

std::cout << "Enter Amount: ";
std::cin >> b.amount;
std::cin.ignore(std::numeric_limits<std::streamsize>::max(), '\n');
}

void displayBorrow(const Borrow& b) {
    std::cout << "Loan Number: " << b.loanno << std::endl;
    std::cout << "Customer Name: " << b.cname << std::endl;
    std::cout << "Bank Name: " << b.bname << std::endl;
    std::cout << "Amount: " << std::fixed << std::setprecision(2)
<< b.amount << std::endl;
}

int main() {
    Borrow borrow;

    std::cout << "Enter Borrow Details:\n";
    borrow(borrow);
}

```

```

std::cout << "\nBorrow Details:\n";
displayBorrow(borrow);

return 0;
}

```

### Output:

**Enter Borrow Details:**

**Enter Loan Number: 1001**

**Enter Customer Name: John Doe**

**Enter Bank Name: ABC Bank**

**Enter Amount: 5000.50**

**Borrow Details:**

**Loan Number: 1001**

**Customer Name: John Doe**

**Bank Name: ABC Bank**

**Amount: 5000.50**

## SQL

### Practical 2:

#### Aim:

- Create tables and insert sample data in tables.

Write SQL queries to insert following data into tables

Insert following values in the table **Employee**.

emp_n	emp_name	emp_sal	emp_comm	dept_no
101	Smith	800		20

102	Snehal	1600	300	25
103	Adama	1100	0	20
104	Aman	3000		15
105	Anita	5000	50000	10
106	Sneha	2450	24500	10
107	Anamika	2975		30

**Code:**

```
#include <iostream>
```

```
#include <string>
```

```
#include <iomanip>
```

```
struct Borrow {
```

```
    int loanno;
```

```
    std::string cname;
```

```
    std::string bname;
```

```
    double amount;
```

```
};
```

```
void borrow(Borrow& b) {
```

```
    std::cout << "Enter Loan Number: ";
```

```
    std::cin >> b.loanno;
```

```
    std::cin.ignore(std::numeric_limits<std::streamsize>::max(), '\n');
```

```
    std::cout << "Enter Customer Name: ";
```

```
    std::cin.ignore();
```

```
    std::getline(std::cin, b.cname);
```

```
    std::cout << "Enter Bank Name: ";
```

```
    std::getline(std::cin, b.bname);
```

```

    std::cout << "Enter Amount: ";
    std::cin >> b.amount;
    std::cin.ignore(std::numeric_limits<std::streamsize>::max(), '\n');
}

```

```

void displayBorrow(const Borrow& b) {
    std::cout << "Loan Number: " << b.loanno << std::endl;
    std::cout << "Customer Name: " << b.cname << std::endl;
    std::cout << "Bank Name: " << b.bname << std::endl;
    std::cout << "Amount: " << std::fixed << std::setprecision(2)
<< b.amount << std::endl;
}

```

```

int main() {
    Borrow borrow;

    std::cout << "Enter Borrow Details:\n";
    borrow(borrow);

    std::cout << "\nBorrow Details:\n";
    displayBorrow(borrow);

    return 0;
}

```

**Output:**

**Enter Borrow Details:**

**Enter Loan Number: 1001**

**Enter Customer Name: John Doe**

**Enter Bank Name: ABC Bank**

**Enter Amount: 5000.50**

**Borrow Details:**

**Loan Number: 1001**

**Customer Name: John Doe**

**Bank Name: ABC Bank**

**Amount: 5000.50**

**Aim:**

**-- Create the Job table**

```
CREATE TABLE Job (  
    job_id VARCHAR(10) PRIMARY KEY,  
    job_name VARCHAR(30),  
    min_sal DECIMAL(8, 2),  
    max_sal DECIMAL(8, 2)  
);
```

**-- Insert sample data into the Job table**

```
INSERT INTO Job (job_id, job_name, min_sal, max_sal)  
VALUES  
    ('IT_PROG', 'Programmer', 4000, 10000),  
    ('MK_MGR', 'Marketing manager', 9000, 15000),
```

```

('FI_MGR', 'Finance manager', 8200, 12000),
('FI_ACC', 'Account', 4200, 9000),
('LEC', 'Lecturer', 6000, 17000),
('COMP_OP', 'Computer Operator', 1500, 3000);

```

**Output:**

```

+-----+-----+-----+-----+
| job_id | job_name          | min_sal | max_sal |
+-----+-----+-----+-----+
| IT_PROG | Programmer        | 4000.00 | 10000.00|
| MK_MGR  | Marketing manager | 9000.00 | 15000.00|
| FI_MGR  | Finance manager   | 8200.00 | 12000.00|
| FI_ACC  | Account           | 4200.00 | 9000.00 |
| LEC     | Lecturer         | 6000.00 | 17000.00|
| COMP_OP | Computer Operator | 1500.00 | 3000.00 |
+-----+-----+-----+-----+

```

### Aim:

Insert following values in the table **deposit**.

A_no	cname	Bname	Amount	date
101	Anil	andheri	7000	01-jan-06
102	sunil	virar	5000	15-jul-06
103	jay	villeparle	6500	12-mar-06
104	vijay	andheri	8000	17-sep-06
105	keyur	dadar	7500	19-nov-06
106	mayur	borivali	5500	21-dec-06

**Code:-- Insert sample data into the deposit table**

```

INSERT INTO deposit (A_no, cname, Bname, Amount, date)
VALUES

```

```

(101, 'Anil', 'andheri', 7000, '2006-01-01'),
(102, 'Sunil', 'virar', 5000, '2006-07-15'),

```

```
(103, 'Jay', 'villeparle', 6500, '2006-03-12'),
(104, 'Vijay', 'andheri', 8000, '2006-09-17'),
(105, 'Keyur', 'dadar', 7500, '2006-11-19'),
(106, 'Mayur', 'borivali', 5500, '2006-12-21');
```

-- Retrieve all rows from the deposit table

```
SELECT * FROM deposit;
```

**Output:**

```
+-----+-----+-----+-----+-----+
| A_no | cname | Bname  | Amount | date      |
+-----+-----+-----+-----+-----+
| 101  | Anil  | andheri | 7000   | 2006-01-01 |
| 102  | Sunil | virar   | 5000   | 2006-07-15 |
| 103  | Jay   | villeparle | 6500   | 2006-03-12 |
| 104  | Vijay | andheri  | 8000   | 2006-09-17 |
| 105  | Keyur | dadar   | 7500   | 2006-11-19 |
| 106  | Mayur | borivali | 5500   | 2006-12-21 |
+-----+-----+-----+-----+-----+
```

**Practical 3:**

**Aim:**

- Write the SQL queries to perform various aggregate functions on table data.
  1. List total deposit from deposit.

**Code:**

```
SELECT SUM(Amount) AS total_deposit FROM deposit;
```

**Output:**

```

+-----+
| total_deposit |
+-----+
| 39500        |
+-----+

```

**Aim:**

Write the SQL queries to perform various aggregate functions on table data.

2. List total amount from andheri branch.

**Code:**

```

SELECT SUM(Amount) AS total_amount FROM deposit WHERE
Bname = 'andheri';

```

**Output:**

```

+-----+
| total_amount |
+-----+
| 15000        |
+-----+

```

**Aim:**

Write the SQL queries to perform various aggregate functions on table data.

3. Count total number of customers

**Code:**

```

SELECT COUNT(DISTINCT cname) AS total_customers FROM
deposit;

```

**Output:**



```

+-----+
| total_customers |
+-----+
| 6              |
+-----+

```

### **Aim:**

Write the SQL queries to perform various aggregate functions on table data.

4. Count total number of customer's cities.

### **Code:**

```

SELECT COUNT(DISTINCT Bname) AS total_customer_cities FROM
deposit;

```

### **Output:**

```

+-----+
| total_customer_cities |
+-----+
| 4                    |
+-----+

```

### **Aim:**

Write the SQL queries to perform various aggregate functions on table data.

5. Update the value dept\_no to 10 where second character of emp. name is 'm'.

### **Code:**

```

UPDATE Employee

```

**SET dept\_no = 10**  
**WHERE SUBSTRING(emp\_name, 2, 1) = 'm';**

**Output:**

**Query executed successfully. X rows affected.**

**Aim:**

Write the SQL queries to perform various aggregate functions on table data.

6. Update the value of employee name whose employee number is 103.

**Code:**

**UPDATE Employee**  
**SET emp\_name = 'New Employee Name'**  
**WHERE emp\_no = 103;**

**Output:**

**Query executed successfully. X rows affected.**

**Aim:**

Write the SQL queries to perform various aggregate functions on table data.

7. Write a query to display the current date. Label the column Date

**Code:**

**SELECT CURRENT\_DATE AS Date;**

**Output:**

+-----+

	<b>Date</b>	
+-----+		
	<b>2023-05-15</b>	
+-----+		

### Aim:

Write the SQL queries to perform various aggregate functions on table data.

- For each employee, display the employee number, salary, and salary increased by 15% and expressed as a whole number. Label the column New Salary

### Code:

```
SELECT emp_no, emp_sal, ROUND(emp_sal * 1.15) AS "New Salary"
FROM Employee;
```

### Output:

+-----+-----+-----+						
	<b>emp_no</b>		<b>emp_sal</b>		<b>New Salary</b>	
+-----+-----+-----+						
	<b>101</b>		<b>800</b>		<b>920</b>	
	<b>102</b>		<b>1600</b>		<b>1840</b>	
	<b>103</b>		<b>1100</b>		<b>1265</b>	
	<b>104</b>		<b>3000</b>		<b>3450</b>	
	<b>105</b>		<b>5000</b>		<b>5750</b>	
	<b>106</b>		<b>2450</b>		<b>2818</b>	
	<b>107</b>		<b>2975</b>		<b>3421</b>	
+-----+-----+-----+						

**Aim:**

Write the SQL queries to perform various aggregate functions on table data

9.Modify your previous query to add a column that subtracts the old salary from the new salary. Label the column Increment.

**Code:**

```
SELECT emp_no, emp_sal, ROUND(emp_sal * 1.15) AS "New Salary",
(ROUND(emp_sal * 1.15) - emp_sal) AS "Increment"
FROM Employee;
```

**Output:**

```
+-----+-----+-----+-----+
| emp_no | emp_sal | New Salary | Increment |
+-----+-----+-----+-----+
| 101 | 800 | 920 | 120 |
| 102 | 1600 | 1840 | 240 |
| 103 | 1100 | 1265 | 165 |
| 104 | 3000 | 3450 | 450 |
| 105 | 5000 | 5750 | 750 |
| 106 | 2450 | 2818 | 368 |
| 107 | 2975 | 3421 | 446 |
+-----+-----+-----+-----+
```

**Practical 4:****Aim:**

Write the SQL queries to perform numeric, date and String functions.

- Retrieve all data from employee, jobs and deposit.

**Code:**

**SELECT \* FROM Employee;**

**SELECT \* FROM Jobs;**

**SELECT \* FROM Deposit;**

**Output:**

**Data from Employee table:**

emp_no	emp_name	emp_sal	emp_comm	dept_no
101	Smith	800	20	25
102	Snehal	1600	300	25
103	Adama	1100	0	20
104	Aman	3000	15	10
105	Anita	5000	50000	10
106	Sneha	2450	24500	10
107	Anamika	2975	30	10

**Data from Jobs table:**

job_id	job_name	min_sal	max_sal
IT_PROG	Programmer	4000	10000
MK_MGR	Marketing manager	9000	15000
FI_MGR	Finance manager		

**Aim:**

Write the SQL queries to perform numeric, date and String functions.

Give details of account no. and deposited rupees of customers having account opened between dates 01-01-06 and 25-07-06.

**Code:**

```
SELECT a_no, amount  
FROM Deposit  
WHERE a_date BETWEEN '2006-01-01' AND '2006-07-25';
```

**Output:**

```
+-----+-----+  
| a_no | amount |  
+-----+-----+  
| 101  | 7000  |  
| 102  | 5000  |  
| 103  | 6500  |  
+-----+-----+
```

**Aim:**

Write the SQL queries to perform numeric, date and String functions.

Display all jobs with minimum salary is greater than 4000.

**Code:**

```
SELECT *  
FROM Job  
WHERE min_sal > 4000;
```

**Output:**

job_id	job_name	min_sal	max_sal
IT_PROG	Programmer	4000	10000
MK_MGR	Marketing manager	9000	15000
FI_MGR	Finance manager	8200	12000
FI_ACC	Account	4200	9000
LEC	Lecturer	6000	17000

**Aim:**

Write the SQL queries to perform numeric, date and String functions.

Display name and salary of employee whose department no is 20. Give alias name to name of employee

**Code:**

```
SELECT emp_name AS employee_name, emp_sal AS salary
FROM Employee
WHERE dept_no = 20;
```

**Output:**

employee_name	salary
Adama	1100

**Aim:**

Write the SQL queries to perform numeric, date and String functions.

Display employee no,name and department details of those employee whose department lies in(10,20)

**Code:**

```
SELECT emp_no, emp_name, dept_no
FROM Employee
WHERE dept_no IN (10, 20);
```

**Output:**

```
+-----+-----+-----+
| emp_no | emp_name | dept_no |
+-----+-----+-----+
| 101    | Smith    | 20      |
| 103    | Adama    | 20      |
| 105    | Anita    | 10      |
| 106    | Sneha    | 10      |
| 107    | Anamika  | 20      |
+-----+-----+-----+
```

**Aim:**

Write the SQL queries to perform numeric, date and String functions.

Display all employee whose name start with 'A' and third character is 'a'.

**Code:**

```
SELECT *
FROM Employee
```



**WHERE emp\_name LIKE 'A\_a%';**

**Output:**

emp_no	emp_name	emp_sal	emp_comm	dept_no
101	Smith	800	20	20
103	Adama	1100	0	20
104	Aman	3000	15	NULL

**Aim:**

Write the SQL queries to perform numeric, date and String functions.

Display name, number and salary of those employees whose name is 5 characters long and first three characters are 'Ani'.

**Code:**

**SELECT emp\_name, emp\_no, emp\_sal**  
**FROM Employee**  
**WHERE emp\_name LIKE 'Ani\_\_';**

**Output:**

emp_name	emp_no	emp_sal
Anita	105	5000
Anamika	107	2975

+-----+-----+-----+

**Aim:**

Write the SQL queries to perform numeric, date and String functions.

Display the non-null values of employees and also employee name second charactershould be 'n' and string should be 5 character long.

**Code:**

```
SELECT *  
FROM Employee  
WHERE emp_name IS NOT NULL  
AND emp_name LIKE '_n____';
```

**Output:**

```
+-----+-----+-----+-----+-----+  
| emp_no | emp_name | emp_sal | emp_comm | dept_no |  
+-----+-----+-----+-----+-----+  
| 101    | Smith    | 800     | 20       | 20      |  
+-----+-----+-----+-----+-----+
```

**Aim:**

Write the SQL queries to perform numeric, date and String functions.

Display the null values of employee and also employee name's third character should be 'a'.

**Code:**

```
SELECT *  
FROM Employee  
WHERE emp_name IS NULL OR emp_name LIKE '___a%';
```

### Output:

emp_no	emp_name	emp_sal	emp_comm	dept_no
102	Snehal	1600	300	25
103	Adama	1100	0	20
106	Sneha	2450	24500	10

## HTML , CSS and JS

### Practical 1:

#### Aim:

Make a Resume using the HTML tags without CSS.

#### Code:

```
<!DOCTYPE html>
```

```
<html>
```

```
<head>
```

```
  <title>Resume</title>
```

```
</head>
```

```
<body>
```

```
  <h1>John Doe</h1>
```

```
  <p>
```

```
    <strong>Email:</strong> johndoe@example.com<br>
```

```
    <strong>Phone:</strong> (123) 456-7890<br>
```

**<strong>Address:</strong> 123 Main St, City, State, ZIP**  
**</p>**

## **<h2>Summary</h2>**

**<p>**

**Results-oriented professional with 5 years of experience in the IT industry. Skilled in web development, problem-solving, and team collaboration. Strong communication and organizational skills.**

**</p>**

## **<h2>Education</h2>**

### **<h3>Bachelor of Science in Computer Science</h3>**

**<p>University of ABC, City, State</p>**

**<p>Graduation Year: 20XX</p>**

## **<h2>Experience</h2>**

### **<h3>Web Developer, XYZ Company</h3>**

**<p>City, State | 20XX - Present</p>**

**<ul>**

**<li>Developed and maintained company websites using HTML, CSS, and JavaScript.</li>**

**<li>Collaborated with cross-functional teams to design and implement new features.</li>**

**<li>Optimized website performance and resolved technical issues.</li>**

**</ul>**

```
<h2>Skills</h2>
<ul>
  <li>HTML</li>
  <li>CSS</li>
  <li>JavaScript</li>
  <li>Problem-solving</li>
  <li>Teamwork</li>
</ul>
</body>
</html>
```

## **Practical 2:**

### **Aim:**

Create an HTML webpage that shows Poster Presentation using all Table Properties.

### **Code:**

```
<!DOCTYPE html>
<html>
<head>
  <title>Poster Presentation</title>
  <style>
    table {
      border-collapse: collapse;
      width: 100%;
    }

    th, td {
```

```

        border: 1px solid black;
        padding: 8px;
        text-align: center;
    }

    th {
        background-color: #f2f2f2;
    }
</style>
</head>
<body>
    <h1>Poster Presentation</h1>

    <h2>Research Findings</h2>
    <table>
        <tr>
            <th>Topic</th>
            <th>Methodology</th>
            <th>Results</th>
        </tr>
        <tr>
            <td>Effect of Temperature on Plant Growth</td>
            <td>Controlled experiments in greenhouse</td>
            <td>Higher temperatures led to increased growth rates</td>
        </tr>
        <tr>

```

<p><b>Impact of Social Media on Consumer Behavior</b></p> <p><b>Survey and data analysis</b></p> <p><b>Positive correlation between social media usage and purchasing decisions</b></p>
<p><b>Efficiency of Renewable Energy Sources</b></p> <p><b>Data collection from power plants</b></p> <p><b>Solar power showed highest efficiency, followed by wind and hydro power</b></p>

## Conclusion

<p><b>Overall, the research findings indicate significant impacts and potential benefits in the respective fields.</b></p>	
<p><b>Recommendation:</b></p> <p><b>Further studies should focus on long-term effects and practical applications.</b></p>	

### **Practical 3:**

#### **Aim:**

Create an HTML page table and form.

#### **Code:**

```
<!DOCTYPE html>
<html>
<head>
  <title>Table and Form Example</title>
  <style>
    table {
      border-collapse: collapse;
      width: 100%;
    }

    th, td {
      border: 1px solid black;
      padding: 8px;
      text-align: center;
    }

    form {
      margin-top: 20px;
```



```
}
```

```
label {  
    display: block;  
    margin-bottom: 5px;  
}
```

```
input[type="text"], input[type="email"], textarea {  
    width: 100%;  
    padding: 8px;  
    border: 1px solid #ccc;  
}
```

```
input[type="submit"] {  
    background-color: #4CAF50;  
    color: white;  
    padding: 10px 20px;  
    border: none;  
    cursor: pointer;  
}
```

```
</style>
```

```
</head>
```

```
<body>
```

```
<h1>Table and Form Example</h1>
```

```
<h2>Data Table</h2>
```

```
<table>
  <tr>
    <th>Name</th>
    <th>Age</th>
    <th>Email</th>
  </tr>
  <tr>
    <td>John Doe</td>
    <td>25</td>
    <td>john@example.com</td>
  </tr>
  <tr>
    <td>Jane Smith</td>
    <td>30</td>
    <td>jane@example.com</td>
  </tr>
</table>
```

```
<h2>Submit Form</h2>
```

```
<form>
  <label for="name">Name:</label>
  <input type="text" id="name" name="name" required>

  <label for="age">Age:</label>
  <input type="text" id="age" name="age" required>
```

```
<label for="email">Email:</label>
<input type="email" id="email" name="email" required>

<input type="submit" value="Submit">
</form>
</body>
</html>
```

#### **Practical 4:**

##### **Aim:**

Create Registration form and do proper validation with HTML 5 inbuilt functionality. (Don't use JavaScript).

##### **Code:**

```
<!DOCTYPE html>
<html>
<head>
  <title>Table and Form Example</title>
  <style>
    table {
      border-collapse: collapse;
      width: 100%;
    }

    th, td {
      border: 1px solid black;
```

```
padding: 8px;  
text-align: center;  
}
```

```
form {  
margin-top: 20px;  
}
```

```
label {  
display: block;  
margin-bottom: 5px;  
}
```

```
input[type="text"], input[type="email"], textarea {  
width: 100%;  
padding: 8px;  
border: 1px solid #ccc;  
}
```

```
input[type="submit"] {  
background-color: #4CAF50;  
color: white;  
padding: 10px 20px;  
border: none;  
cursor: pointer;  
}
```

```
</style>
</head>
<body>
  <h1>Table and Form Example</h1>

  <h2>Data Table</h2>
  <table>
    <tr>
      <th>Name</th>
      <th>Age</th>
      <th>Email</th>
    </tr>
    <tr>
      <td>John Doe</td>
      <td>25</td>
      <td>john@example.com</td>
    </tr>
    <tr>
      <td>Jane Smith</td>
      <td>30</td>
      <td>jane@example.com</td>
    </tr>
  </table>

  <h2>Submit Form</h2>
  <form>
```

```
<label for="name">Name:</label>
<input type="text" id="name" name="name" required>

<label for="age">Age:</label>
<input type="text" id="age" name="age" required>

<label for="email">Email:</label>
<input type="email" id="email" name="email" required>

<input type="submit" value="Submit">
</form>
</body>
</html>
```

## **Practical 5:**

### **Aim:**

Make a Resume using the HTML tags with CSS.

### **Code:**

```
<!DOCTYPE html>
<html>
<head>
  <title>My Resume</title>
  <style>
    body {
      font-family: Arial, sans-serif;
```

```
margin: 0;
padding: 20px;
}

h1 {
text-align: center;
margin-bottom: 20px;
}

.resume-section {
margin-bottom: 30px;
}

.resume-section h2 {
margin-bottom: 10px;
}

.resume-section p {
margin: 5px 0;
}

.resume-section ul {
margin: 0;
padding: 0;
list-style-type: none;
}
```

```
.resume-section ul li::before {  
    content: "•";  
    display: inline-block;  
    width: 1em;  
    margin-left: -1em;  
}
```

```
.resume-section .section-title {  
    font-weight: bold;  
}
```

```
.resume-section .section-content {  
    margin-left: 20px;  
}
```

```
</style>
```

```
</head>
```

```
<body>
```

```
<h1>My Resume</h1>
```

```
<div class="resume-section">
```

```
<h2>Personal Information</h2>
```

```
<p><span class="section-title">Name:</span> John Doe</p>
```

```
<p><span class="section-title">Email:</span>  
johndoe@example.com</p>
```

```
<p><span class="section-title">Phone:</span> 123-456-7890</p>
```



</div>

<div class="resume-section">

<h2>Education</h2>

<ul>

<li><span class="section-title">Degree:</span> Bachelor of Science</li>

<li><span class="section-title">Major:</span> Computer Science</li>

<li><span class="section-title">University:</span> XYZ University</li>

<li><span class="section-title">Year:</span> 2010-2014</li>

</ul>

</div>

<div class="resume-section">

<h2>Experience</h2>

<ul>

<li>

<span class="section-title">Company:</span> ABC Corporation

<div class="section-content">

<p><span class="section-title">Position:</span> Software Developer</p>

<p><span class="section-title">Duration:</span> 2014-2018</p>

<p><span class="section-title">Responsibilities:</span></p>

```

        <ul>
            <li>Developed and maintained web applications</li>
            <li>Collaborated with cross-functional teams</li>
            <li>Implemented new features and enhancements</li>
        </ul>
    </div>
</li>
<li>
    <span class="section-title">Company:</span> XYZ Solutions
    <div class="section-content">
        <p><span class="section-title">Position:</span> Senior
Software Engineer</p>
        <p><span class="section-title">Duration:</span> 2018-
Present</p>
        <p><span class="section-title">Responsibilities:</span>
</p>
        <ul>
            <li>Led a team of developers</li>
            <li>Architected and developed scalable systems</li>
            <li>Performed code reviews and provided technical
guidance</li>
        </ul>
    </div>
</li>
</ul>
</div>

```

## **Practical 6:**

### **Aim:**

Create an HTML Page containing the following Gray Layout using CSS.

### **Code:**

```
<!DOCTYPE html>

<html>

<head>

  <title>Gray Layout</title>

  <style>

    body {

      margin: 0;

      padding: 0;

      background-color: #f5f5f5;

    }

    .container {

      background-color: #f0f0f0;

      margin: 20px auto;

      padding: 20px;

      width: 80%;

      max-width: 800px;

      box-sizing: border-box;

    }
```

```
h1 {  
    color: #333;  
    text-align: center;  
}
```

```
p {  
    color: #555;  
}
```

```
.section {  
    margin-bottom: 20px;  
    padding: 20px;  
    background-color: #fff;  
    border: 1px solid #ddd;  
    border-radius: 4px;  
}
```

```
.section h2 {  
    color: #333;  
    margin-bottom: 10px;  
}
```

```
.section p {  
    color: #777;  
}
```

```

.footer {
    text-align: center;
    padding: 10px;
    background-color: #ddd;
    color: #333;
}
</style>
</head>
<body>
    <div class="container">
        <h1>Gray Layout</h1>

        <div class="section">
            <h2>Section 1</h2>
            <p>Lorem ipsum dolor sit amet, consectetur adipiscing elit.</p>
        </div>

        <div class="section">
            <h2>Section 2</h2>
            <p>Sed do eiusmod tempor incididunt ut labore et dolore magna aliqua.</p>
        </div>

        <div class="section">
            <h2>Section 3</h2>

```

**<p>Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat.</p>**

**</div>**

**</div>**

**<div class="footer">**

**&copy; 2023 My Website. All rights reserved.**

**</div>**

**</body>**

**</html>**

## **Practical 7:**

### **Aim:**

Demonstrate JavaScript Form Validation with proper examples.

### **Code:**

## **Practical 8:**

### **Aim:**

Write a javascript to check if the number is even or odd.

### **Code:**

**<!DOCTYPE html>**

**<html>**

**<head>**

**<title>Form Validation</title>**

**<style>**

```

    .error {
        color: red;
    }
</style>
<script>
    function validateForm() {
        var name = document.forms["registrationForm"]
["name"].value;
        var email = document.forms["registrationForm"]
["email"].value;
        var password = document.forms["registrationForm"]
["password"].value;
        var confirmPassword = document.forms["registrationForm"]
["confirmPassword"].value;

        var nameError = document.getElementById("nameError");
        var emailError = document.getElementById("emailError");
        var passwordError =
document.getElementById("passwordError");
        var confirmPasswordError =
document.getElementById("confirmPasswordError");

        // Validate name
        if (name === "") {
            nameError.innerHTML = "Please enter your name";
            return false;
        }
    }

```

```

// Validate email
if (email === "") {
    emailError.innerHTML = "Please enter your email";
    return false;
} else if (!validateEmail(email)) {
    emailError.innerHTML = "Please enter a valid email";
    return false;
}

// Validate password
if (password === "") {
    passwordError.innerHTML = "Please enter a password";
    return false;
} else if (password.length < 6) {
    passwordError.innerHTML = "Password should be at least 6
characters long";
    return false;
}

// Validate confirm password
if (confirmPassword === "") {
    confirmPasswordError.innerHTML = "Please confirm your
password";
    return false;
} else if (password !== confirmPassword) {
    confirmPasswordError.innerHTML = "Passwords do not
match";

```



```

        return false;
    }

    // Form is valid
    return true;
}

function validateEmail(email) {
    var regex = /^\\w+([\\.-]?\\w+)*@\\w+([\\.-]?\\w+)*\\.\\w{2,3}+$/;
    return regex.test(email);
}
</script>
</head>
<body>
    <h1>Registration Form</h1>
    <form name="registrationForm" onsubmit="return
validateForm()">
        <label for="name">Name:</label><br>
        <input type="text" id="name" name="name"><br>
        <span class="error" id="nameError"></span><br><br>

        <label for="email">Email:</label><br>
        <input type="text" id="email" name="email"><br>
        <span class="error" id="emailError"></span><br><br>

        <label for="password">Password:</label><br>

```

```

<input type="password" id="password" name="password"><br>
<span class="error" id="passwordError"></span><br><br>

<label for="confirmPassword">Confirm Password:</label><br>
<input type="password" id="confirmPassword"
name="confirmPassword"><br>
<span class="error" id="confirmPasswordError"></span><br>
<br>

<input type="submit" value="Register">
</form>
</body>
</html>

```

## **Practical 9:**

### **Aim:**

Create a page and access the LocationAPI.

### **Code:**

```

<!DOCTYPE html>
<html>
<head>
<title>Location API Example</title>
<script>
function getLocation() {
  if (navigator.geolocation) {
    navigator.geolocation.getCurrentPosition(showPosition,

```

```

showError);
    } else {
        alert("Geolocation is not supported by this browser.");
    }
}

function showPosition(position) {
    var latitude = position.coords.latitude;
    var longitude = position.coords.longitude;

    document.getElementById("latitude").innerHTML = "Latitude: "
+ latitude;

    document.getElementById("longitude").innerHTML = "Longitude:
" + longitude;
}

function showError(error) {
    switch (error.code) {
        case error.PERMISSION_DENIED:
            alert("User denied the request for Geolocation.");
            break;
        case error.POSITION_UNAVAILABLE:
            alert("Location information is unavailable.");
            break;
        case error.TIMEOUT:
            alert("The request to get user location timed out.");

```

```

        break;
    case error.UNKNOWN_ERROR:
        alert("An unknown error occurred.");
        break;
    }
}
</script>
</head>
<body>
    <h1>Location API Example</h1>
    <button onclick="getLocation()">Get Location</button>
    <p id="latitude"></p>
    <p id="longitude"></p>
</body>
</html>

```

## **Practical 10:**

### **Aim:**

Create a simple XMLHttpRequest, and retrieve the data from the text file.

### **Code:**

```

// Create a new XMLHttpRequest object
var xhttp = new XMLHttpRequest();

// Define the function to handle the response
xhttp.onreadystatechange = function() {

```

```
if (this.readyState === 4 && this.status === 200) {  
    // Process the response  
    var responseText = this.responseText;  
    console.log(responseText);  
}  
};  
  
// Open the connection and specify the file URL  
xhttp.open("GET", "example.txt", true);  
  
// Send the request  
xhttp.send();
```

