\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Data Structure Lab

CEN-391

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Program 5

Code :-

#include <iostream>

#include <string.h>

using namespace std;

struct Employee

{

    int Eid;

    char Name[30];

    float Salary;

    struct Employee \*next;

};

void Add\_Employee(Employee \*&Emp\_Data, int &size)

{

    cout << "Add Employee..." << endl;

    struct Employee \*newEmployee = (Employee \*)malloc(sizeof(Employee));

    int Eid;

    bool check = false;

    do

    {

        cout << "Enter The Employee Eid : ";

        cin >> Eid;

        Employee\*temp=Emp\_Data;

        while(temp!=nullptr)

        {

            if (Eid == temp->Eid)

            {

                cout << endl

                     << "Eid Already Exist!" << endl;

                cout << "Try Again!" << endl

                     << endl;

                check = true;

            }

            temp=temp->next;

        }

    } while (check);

    newEmployee->Eid = Eid;

    fflush(stdin);

    cout << "Enter The Employee Name : ";

    gets(newEmployee->Name);

    cout << "Enter The Employee Salary : ";

    cin >> newEmployee->Salary;

    newEmployee->next = Emp\_Data;

    Emp\_Data = newEmployee;

    size++;

}

void Display\_Employee(Employee \*Emp\_Data, int &size)

{

    if (size == 0)

    {

        cout << endl

             << "Empty!" << endl;

        return;

    }

    cout <<  "Display All Employee..." << endl;

    cout << "|Eid\t\t|"

         << "Name\t\t|"

         << "Salary\t\t|" << endl;

    Employee \*temp = Emp\_Data;

    while (temp != nullptr)

    {

        cout << "\t" << temp->Eid << "\t";

        cout << temp->Name << "\t";

        cout << temp->Salary << "\t" << endl;

        temp = temp->next;

    }

}

void Search\_Employee\_Eid(Employee \*Emp\_Data, int &size)

{

    cout << "Search Employee By Eid..." << endl;

    if (size == 0)

    {

        cout << "Empty!" << endl;

        return;

    }

    int Eid;

    cout << "Enter The Employee Eid : ";

    cin >> Eid;

    cout << endl;

    Employee \*temp = Emp\_Data;

    while (temp != nullptr)

    {

        if (temp->Eid == Eid)

        {

            cout << "Employee Found!\n\nDetails..." << endl;

            cout << "Eid : " << temp->Eid << "\t  ";

            cout << "Name : " << temp->Name << "\t  ";

            cout << "Salary : " << temp->Salary << endl;

            break;

        }

        temp = temp->next;

    }

    if (temp == nullptr)

    {

        cout << "Employee Not Found!" << endl;

    }

}

void Search\_Employee\_Name(Employee \*Emp\_Data, int &size)

{

    cout << "Search Employee By Name..." << endl;

    if (size == 0)

    {

        cout << "Empty!" << endl;

        return;

    }

    char Name[30];

    cout << "Enter The Name Of Your Employee : ";

    fflush(stdin);

    gets(Name);

    cout << endl;

    Employee \*temp = Emp\_Data;

    while (temp != nullptr)

    {

        if (!strcmp(Name, temp->Name))

        {

            cout << "Employee Found!\n\nDetails..." << endl;

            cout << "Eid : " << temp->Eid << "\t  ";

            cout << "Name : " << temp->Name << "\t  ";

            cout << "Salary : " << temp->Salary << endl;

            break;

        }

        temp = temp->next;

    }

    if (temp == nullptr)

    {

        cout << "Employee Not Found!" << endl;

    }

}

void Highest\_Salary(Employee \*Emp\_Data, int &size)

{

    cout << "Highest Salary Of Employee" << endl;

    if (size == 0)

    {

        cout << "Empty!" << endl;

        return;

    }

    Employee \*temp = Emp\_Data->next, \*MaxEmployee = Emp\_Data;

    while (temp != nullptr)

    {

        if (MaxEmployee->Salary < temp->Salary)

        {

            MaxEmployee = temp;

        }

        temp=temp->next;

    }

    temp = Emp\_Data;

    while (temp != nullptr)

    {

        if (MaxEmployee->Salary == temp->Salary)

        {

            cout << "Eid : " << temp->Eid << "\t  ";

            cout << "Name : " << temp->Name << "\t ";

            cout << "Salary : " << temp->Salary << endl;

        }

        temp=temp->next;

    }

}

void Total\_Employee(int &size)

{

    cout << endl

         << "No Of Employee..." << endl;

    cout << endl

         << "Total No Of Employee : ";

    cout << size << endl;

}

void AnsBar()

{

    cout<<"-------------------------------------------------------------------------------\n";

}

void Menu()

{

    cout << endl

         << endl

         << "\_\_\_Operations\_\_\_" << endl;

    cout << "1.Add Employee" << endl;

    cout << "2.Display Employee" << endl;

    cout << "3.Search Employee Byy Eid" << endl;

    cout << "4.Search Employee By Name" << endl;

    cout << "5.Employee having Higest Salary" << endl;

    cout << "6.Total No Of Employee" << endl;

    cout << "7.Exit" << endl;

    cout << "Enter Your Choice : ";

}

bool Options(Employee \*&Emp\_Data, int &size)

{

    int opt;

    cin >> opt;

    switch (opt)

    {

    case 1:AnsBar();

        Add\_Employee(Emp\_Data, size);

        break;

    case 2:AnsBar();

        Display\_Employee(Emp\_Data, size);

        break;

    case 3:AnsBar();

        Search\_Employee\_Eid(Emp\_Data, size);

        break;

    case 4:AnsBar();

        Search\_Employee\_Name(Emp\_Data, size);

        break;

    case 5:AnsBar();

        Highest\_Salary(Emp\_Data, size);

        break;

    case 6:AnsBar();

        Total\_Employee(size);

        break;

    case 7:AnsBar();

    cout<<"Exit Operation Is Selected"<<endl;

    AnsBar();

        return 0;

    default:

        cout << "Invalid Input!\nTry Again!" << endl;

    }

    AnsBar();

    return 1;

}

int main()

{

    system("cls");

    cout << "\_\_Vicky Gupta 20BCS070\_\_" << endl;

    struct Employee \*Emp\_Data = nullptr;

    int size = 0;

    while (true)

    {

        Menu();

        if (!Options(Emp\_Data, size))

            break;

    }

    cout << endl

         << "Exiting..." << endl;

    return 0;

}

Output :-

Text

Description automatically generated

Text

Description automatically generated

Text

Description automatically generated

Text

Description automatically generated

Text

Description automatically generated