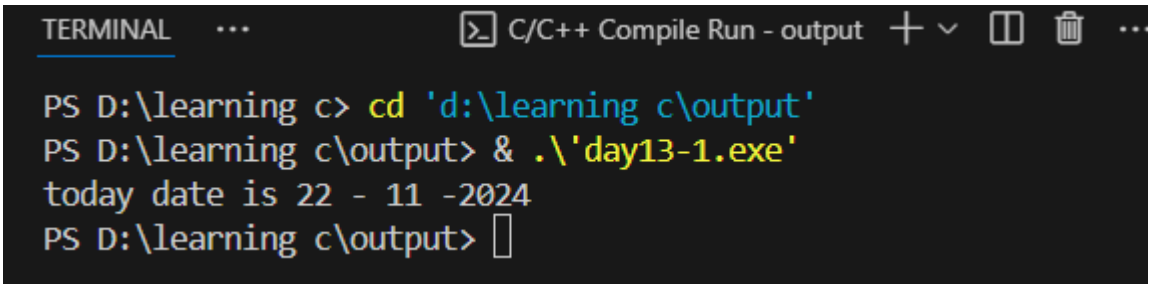
DAY 13

1.//structures and pointers

```
#include<stdio.h>
struct date{
   int day;
   int month;
   int year;
};
int main()
{
   struct date currentDate;
   struct date *sptr;
   sptr=&currentDate;
   (*sptr).day=22;
   (*sptr).month=11;
   (*sptr).year=2024;
   printf("today date is %d - %d -%d",(*sptr).day,(*sptr).month,(*sptr).year);
}
```



2. //using structure pointer operator

```
#include<stdio.h>

struct date{

   int day;

   int month;

   int year;

};

int main()

{

   struct date currentDate;

   struct date *sptr;

   sptr=&currentDate;

   sptr->day=22;

   sptr->month=11;
```

```
    sptr->year=2024;
    printf("today date is %d - %d -%d",sptr->day,sptr->month,sptr->year);
}
```

```
PS D:\learning c\output> cd 'd:\learning c\output'
PS D:\learning c\output> & .\'day13-2.exe'
today date is 22 - 11 -2024
PS D:\learning c\output>
```

3. //structure containing pointers

```
#include<stdio.h>
struct intPts
{
    int *p1;
    int *p2;
};
int main()
{
    struct intPts point;
    int i1=90,i2;
    point.p1=&i1;
    point.p2=&i2;
    *point.p2=89;
    printf("i1=%d , i2=%d",*point.p1,*point.p2);
}
```

```
PS D:\learning c\output> cd 'd:\learning c\output'
PS D:\learning c\output> & .\'day13-3.exe'
i1=90 , i2=89
PS D:\learning c\output>
```

4. //character strings or character pointers

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

```
struct names
{
   char first[40];
   char last[40];
};
struct pnames
{
   char *first;
   char *last;
};
int main()
{
   struct names Cnames={"rinta","ria"};
   struct pnames Pnames={"rinta","ria"};
   printf("%s ,%s ",Cnames.first,Cnames.last);
   printf("size of Cnames =%d",sizeof(Cnames));
   printf("size of Pnames =%d",sizeof(Pnames));//less memory

   return 0;
}
```

```
PS D:\learning c\output> & .\'day13-4.exe'
rinta ,ria size of Cnames =80size of Pnames =8
PS D:\learning c\output>
```

5. //structures as arguments to functions

```
//character strings or character pointers
#include<stdio.h>
#include<string.h>
#include<stdbool.h>
struct names
{
   char first[40];
```

```
    char last[40];
};
bool nameComparison(struct names,struct names);
int main()
{
    struct names Cnames={"rinta","ria"};
    struct names Pnames={"rinta","ria"};
    bool b=nameComparison(Cnames,Pnames);
    printf("b=%d",b);

    return 0;
}
bool nameComparison(struct names Cnames,struct names Pnames)
{
    if(strcmp(Cnames.first,Pnames.first)==0)
    {
        return true;

    }
    else
    {
        return false;
    }
}
```

```
PS D:\learning c> cd 'd:\learning c\output'
PS D:\learning c\output> & .\'day13-5.exe'
b=1
PS D:\learning c\output>
```

6. //pointers to structures as argument

```
#include<stdio.h>
#include<string.h>
#include<stdbool.h>
struct names
{
   char first[40];
   char last[40];
};
bool nameComparison(struct names *,struct names *);
int main()
{
   struct names Cnames={"rinta","ria"};
   struct names Pnames={"rinta","ria"};
   /*struct names *ptr1,*ptr2;
   ptr1=&Cnames;
   ptr2=&Pnames;*/
   bool b=nameComparison(&Cnames,&Pnames);
   printf("b=%d",b);

   return 0;
}
bool nameComparison(struct names *p1,struct names *p2)
{
   if(strcmp(p1->first,p2->first)==0)
   {
      return true;

   }
   else
   {
      return false;
```

```
    }
}
```

```
PS D:\learning c\output> cd 'd:\learning c\output'
PS D:\learning c\output> & .\'day13-6.exe'
b=1
PS D:\learning c\output> cd 'd:\learning c\output'
PS D:\learning c\output> & .\'day13-6.exe'
b=1
PS D:\learning c\output>
```

7. //pointers to structures as argument

```
#include<stdio.h>
#include<string.h>
#include<stdbool.h>
struct names
{
    char first[40];
    char last[40];
};
bool nameComparison(struct names *,struct names *);
int main()
{
    struct names Cnames={"rinta","ria"};
    struct names Pnames={"rinta","ria"};
    struct names *ptr1,*ptr2;
    ptr1=&Cnames;
    ptr2=&Pnames;
    bool b=nameComparison(ptr1,ptr2);
    printf("b=%d",b);
```

```
    return 0;
}
bool nameComparison(struct names *ptr1,struct names *ptr2)
{
    if(strcmp(ptr1->first,ptr2->first)==0)
    {
        return true;


    }
    else
    {
        return false;
    }
}
```

```
PS D:\learning c\output> cd 'd:\learning c\output'
PS D:\learning c\output> & .\'day13-7.exe'
b=1
PS D:\learning c\output>
```

8. **Problem 1: Dynamic Student Record Management**

Objective: Manage student records using pointers to structures and dynamically allocate memory for student names.

Description:

1. Define a structure Student with fields:
    - int roll_no: Roll number
    - char *name: Pointer to dynamically allocated memory for the student's name
    - float marks: Marks obtained
2. Write a program to:
    - Dynamically allocate memory for n students.
    - Accept details of each student, dynamically allocating memory for their names.
    - Display all student details.
    - Free all allocated memory before exiting.

```
//task 1: dyanmic student record management
#include<stdio.h>
#include<string.h>
#include<stdlib.h>
struct students
{
   int roll_no;
   char *name;
   float marks;
};
int main()
{
   int n;

   struct students *std;




   printf("enter the number of students \n");
   scanf("%d",&n);
   std = (struct students *)malloc(n * sizeof(struct students));
   if(std==NULL)
   {
      printf("memory allocation failed \n");

   }
   else
   {
      printf("memory allocated \n");
   }
   for(int i=0;i<n;i++)
```

```
    {
        printf("enter student details \n");
        printf("enter roll number \n");
        scanf("%d",&std[i].roll_no);
        char temp[100];
        printf("enter name \n");
        scanf("%s",temp);
         int l=strlen(temp)+1;
        std[i].name=(char *)malloc(l*sizeof(char));
        if(std[i].name==NULL)
        {
            printf("memory not allocated for name \n");
        }
        else
        {
            printf("memory allocated for name \n");
        }
        strcpy(std[i].name,temp);
        printf("enter marks \n");
        scanf("%f",&std[i].marks);
    }

        printf("student details \n");
        for(int i=0;i<n;i++)
        {
            printf("rollno:%d \n name: %s \n marks:%.2f
\n",std[i].roll_no,std[i].name,std[i].marks);
        }

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

```
    free(std[i].name);
  }
  free(std);

  return 0;
}
```

```
enter the number of students
2
memory allocated
enter student details
enter roll number
1
enter name
rinta
memory allocated for name
enter marks
34
enter student details
enter roll number
2
enter name
ria
memory allocated for name
enter marks
45
student details
rollno:1
 name: rinta
 marks:34.00
rollno:2
 name: ria
 marks:45.00
PS D:\learning c\output>
```

9.Problem 2: Library System with Dynamic Allocation

Objective: Manage a library system where book details are dynamically stored using pointers inside a structure.

Description:

1. Define a structure Book with fields:
    - char *title: Pointer to dynamically allocated memory for the book's title
    - char *author: Pointer to dynamically allocated memory for the author's name
    - int *copies: Pointer to the number of available copies (stored dynamically)
2. Write a program to:
    - Dynamically allocate memory for n books.
    - Accept and display book details.
    - Update the number of copies of a specific book.
    - Free all allocated memory before exiting.

```
//library system dynamic allocation
//task 1: dyanmic student record management
#include<stdio.h>
#include<string.h>
#include<stdlib.h>
struct Book
{
   int *title;
   char *author;
   int copies;
};
int main()
{
   int n;

    struct Book *book;




   printf("enter the number of book \n");
```

```
scanf("%d",&n);
book = (struct Book *)malloc(n * sizeof(struct Book));
if(book==NULL)
{
   printf("memory allocation failed \n");

}
else
{
   printf("memory allocated \n");
}
for(int i=0;i<n;i++)
{
   printf("enter book details \n");
   char temp[100];
   printf("enter title\n");
   scanf("%s",temp);
   int l=strlen(temp)+1;
   book[i].title=(char *)malloc(l*sizeof(char));
   if(book[i].title==NULL)
   {
      printf("memory not allocated for name \n");
   }
   else
   {
      printf("memory allocated for name \n");
   }


   strcpy(book[i].title,temp);
   char auth[100];
```

```
    printf("enter author\n");
    scanf("%s",auth);
    int m=strlen(auth)+1;
    book[i].author=(char *)malloc(m*sizeof(char));
    if(book[i].author==NULL)
    {
        printf("memory not allocated for name \n");
    }
    else
    {
        printf("memory allocated for name \n");
    }
    strcpy(book[i].author,temp);

    printf("enter copies \n");
    scanf("%d",&book[i].copies);
}

    printf("book details \n");
    for(int i=0;i<n;i++)
    {
        printf("title:%s \n author: %s \n copies:%d \n",book[i].title,book[i].author,book[i].copies);
    }

for (int i = 0; i < n; i++)
{
    free(book[i].title);
    free(book[i].author);
 }
 free(book);
```

```
    return 0;
}
```

```
enter the number of book
1
memory allocated
enter book details
enter title
famousfive
memory allocated for name
enter author
agatha
memory allocated for name
enter copies
2
book details
title:famousfive
 author: famousfive
 copies:2
PS D:\learning c\output>
```

10.Problem 1: Complex Number Operations

Objective: Perform addition and multiplication of two complex numbers using structures passed to functions.

Description:

1. Define a structure Complex with fields:
   - float real: Real part of the complex number
   - float imag: Imaginary part of the complex number
2. Write functions to:
   - Add two complex numbers and return the result.
   - Multiply two complex numbers and return the result.
3. Pass the structures as arguments to these functions and display the results.

```
#include<stdio.h>
struct complex
{
    float real;
```

```
    float imag;

};
void add(struct complex, struct complex);
void multiply(struct complex, struct complex);

int main()
{
    struct complex n1;
    struct complex n2;
    printf("number 1 \n");
    printf("enter the real part: \n");
    scanf("%f",&n1.real);
    printf("enter the imaginery part \n");
    scanf("%f",&n1.imag);
    printf("number 2 \n");
    printf("enter the real part: \n");
    scanf("%f",&n2.real);
    printf("enter the imaginery part \n");
    scanf("%f",&n2.imag);
    printf("addition of two complex numbers \n");
    add(n1,n2);
    printf("multiplication of two complex numbers \n");
    multiply(n1,n2);





}
void add(struct complex n1, struct complex n2 )
{
```

```
    struct complex add;
    add.real=n1.real+n2.real;
    add.imag=n1.imag+n2.imag;
    printf("the sum is %.2f +%.2fi \n",add.real,add.imag);



}
void multiply(struct complex n1, struct complex n2)
{
    struct complex multiply;
    multiply.real=n1.real*n2.real;
    multiply.imag=n1.imag*n2.imag;
    printf("the sum is %.2f * %.2fi \n",multiply.real,multiply.imag);



}
```

```
PS D:\learning c> cd 'd:\learning c\output'
PS D:\learning c\output> & .\'day13-10.exe'
number 1
enter the real part:
2
enter the imaginery part
3
number 2
enter the real part:
1
enter the imaginery part
2
addition of two complex numbers
the sum is 3.00 +5.00i
multiplication of two complex numbers
the sum is 2.00 * 6.00i
PS D:\learning c\output>
```

Problem 2: Rectangle Area and Perimeter Calculator

Objective: Calculate the area and perimeter of a rectangle by passing a structure to functions.

Description:

1. Define a structure Rectangle with fields:
    - float length: Length of the rectangle
    - float width: Width of the rectangle
2. Write functions to:
    - Calculate and return the area of the rectangle.
    - Calculate and return the perimeter of the rectangle.
3. Pass the structure to these functions by value and display the results in main.

```
#include <stdio.h>

struct Rectangle {
   float length;
   float width;
};

float area(struct Rectangle);
float perimeter(struct Rectangle);

int main()
{
   struct Rectangle rect;

   printf("Enter the length of the rectangle: ");
   scanf("%f", &rect.length);
   printf("Enter the width of the rectangle: ");
   scanf("%f", &rect.width);
```

```
    printf("Calculating area...\n");
    float a = area(rect);
    printf("Area is %.2f\n", a);

    printf("Calculating perimeter...\n");
    float p = perimeter(rect);
    printf("Perimeter is %.2f\n", p);

    return 0;
}



float area(struct Rectangle rect) {
    return rect.length * rect.width;
}
float perimeter(struct Rectangle rect) {
    return 2 * (rect.length + rect.width);
}
```

```
PS D:\learning c\output> cd 'd:\learning c\output'
PS D:\learning c\output> & .\'day13-11.exe'
Enter the length of the rectangle: 2
Enter the width of the rectangle: 3
Calculating area...
Area is 6.00
Calculating perimeter...
Perimeter is 10.00
PS D:\learning c\output>
```

Problem 3: Student Grade Calculation

Objective: Calculate and assign grades to students based on their marks by passing a structure to a function.

Description:

1. Define a structure Student with fields:

- char name[50]: Name of the student
- int roll_no: Roll number
- float marks[5]: Marks in 5 subjects
- char grade: Grade assigned to the student

2. Write a function to:
   - Calculate the average marks and assign a grade (A, B, etc.) based on predefined criteria.
3. Pass the structure by reference to the function and modify the grade field.

```
#include<stdio.h>
struct Student
{
   char name[50];
   int roll_no;
   float marks[5];
   char grade;
};
void avgMarks(struct Student);
int main()
{
   struct Student std;
   printf("enter the name \n");
   scanf("%s",std.name);
   printf("enter the roll number \n");
   scanf("%d",&std.roll_no);
   printf("enter the marks of 5 subjects \n");
   for(int i=0;i<5;i++)
   {
      scanf("%f",&std.marks[i]);
   }
   avgMarks(std);
```

```
  return 0;
}
void avgMarks(struct Student std)
{
    float sum=0;
    for(int i=0;i<5;i++)
    {
        sum+=std.marks[i];
    }
    float average=sum/5;
    printf("average marks is %.2f \n",average);
    if(average>=90)
    {
        std.grade='A';
    }
    else if(average>=80)
    {
        std.grade='B';
    }
    else if(average>=70)
    {
        std.grade='C';
    }
    else if(average>=60)
    {
        std.grade='D';
    }
    else
    {
        std.grade='F';
    }
```

```
    printf("grade is : %c \n",std.grade);
}
```

```
PS D:\learning c\output> & .\'day13-12.exe'
enter the name
rinta
enter the roll number
2
enter the marks of 5 subjects
45
67
89
90
100
average marks is 78.20
grade is : C
PS D:\learning c\output>
```

Problem 4: Point Operations in 2D Space

Objective: Calculate the distance between two points and check if a point lies within a circle using structures.

Description:

1. Define a structure Point with fields:
    - float x: X-coordinate of the point
    - float y: Y-coordinate of the point
2. Write functions to:
    - Calculate the distance between two points.
    - Check if a given point lies inside a circle of a specified radius (center at origin).
3. Pass the Point structure to these functions and display the results.

```
 #include<stdio.h>
#include<math.h>
struct Point
{
    float x;
    float y;
};
```

```
float distance(struct Point, struct Point);
void circle(struct Point , float);

int main()
{
   struct Point p1;
   struct Point p2;
   printf("point 1 \n");
   printf("enter x \n");
   scanf("%f",&p1.x);
   printf("enter y \n");
   scanf("%f",&p1.y);
   printf("point 2 \n");
   printf("enter x \n");
   scanf("%f",&p2.x);
   printf("enter y \n");
   scanf("%f",&p2.y);

  float d= distance(p1,p2);
   printf("distance between points is %.2f \n",d);
   float r;
   printf("enter the radius of circle \n");
   scanf("%f",&r);
   printf("point 1 \n");
   circle(p1,r);
   printf("point 2 \n");
   circle(p2,r);



   return 0;
```

```
}
float distance(struct Point p1, struct Point p2)
{

    return sqrt(((p1.x -p2.x)*(p1.x-p2.x))+((p1.y-p2.y)*(p1.y-p2.y)));

}
void circle(struct Point p,float r)
{
    float o=sqrt(p.x * p.x + p.y * p.y);
    if(o<=r)
    {
        printf("point inside circle \n");
    }
    else
    {
        printf("point not inside circle \n");
    }
}
```

```
PS D:\learning c\output> cd 'd:\learning c\output'
PS D:\learning c\output> & .\'day13-13.exe'
point 1
enter x
2
enter y
3
point 2
enter x
4
enter y
5
distance between points is 2.83
enter the radius of circle
6
point 1
point inside circle
point 2
point not inside circle
PS D:\learning c\output>
```

Problem 5: Employee Tax Calculation

Objective: Calculate income tax for an employee based on their salary by passing a structure to a function.

Description:

1. Define a structure Employee with fields:
    - char name[50]: Employee name
    - int emp_id: Employee ID
    - float salary: Employee salary
    - float tax: Tax to be calculated (initialized to 0)
2. Write a function to:
    - Calculate tax based on salary slabs (e.g., 10% for salaries below $50,000, 20% otherwise).
    - Modify the tax field of the structure.
3. Pass the structure by reference to the function and display the updated tax in main.

```
#include<stdio.h>

struct Employee
```

```
{
   char name[50];
   int emp_id;
   float salary;
   float tax;

};
void calculateTax(struct Employee *) ;
int main()
{
   struct Employee emp;
   struct Employee *empp;
   empp=&emp;
   printf("enter the name of the employee \n");
   scanf("%s",emp.name);
   printf("enter employee id \n");
   scanf("%d",&emp.emp_id);
   printf("enter the salary \n");
   scanf("%f",&emp.salary);
   emp.tax=0;
   calculateTax(&emp);
   printf("tax is %.2f",emp.tax);

}
void calculateTax(struct Employee *empp)
{
    if(empp->salary<50000)
    {
      empp->tax=empp->salary*0.10;
    }
    else
```

```
    {
      empp->tax=empp->salary*0.20;
    }
}
```

```
PS D:\learning c\output> cd 'd:\learning c\output'
PS D:\learning c\output> & .\'day13-14.exe'
enter the name of the employee
rinta
enter employee id
2
enter the salary
45000
tax is 4500.00
PS D:\learning c\output>
```

15.Problem Statement: Vehicle Service Center Management

Objective: Build a system to manage vehicle servicing records using nested structures.

Description:

1. Define a structure Vehicle with fields:
   - char license_plate[15]: Vehicle's license plate number
   - char owner_name[50]: Owner's name
   - char vehicle_type[20]: Type of vehicle (e.g., car, bike)
2. Define a nested structure Service inside Vehicle with fields:
   - char service_type[30]: Type of service performed
   - float cost: Cost of the service
   - char service_date[12]: Date of service
3. Implement the following features:
   - Add a vehicle to the service center record.
   - Update the service history for a vehicle.
   - Display the service details of a specific vehicle.
   - Generate and display a summary report of all vehicles serviced, including total revenue.

```
#include <stdio.h>

#include <string.h>
```

```
struct Service {
    char service_type[30];
    float cost;
    char service_date[12];
};

struct Vehicle {
    char license_plate[15];
    char owner_name[50];
    char vehicle_type[20];
    struct Service service_history[5];
};

int main() {
    struct Vehicle vehicles[5];
    int vehicle_count = 0;
    int choice;

    while (1) {
        printf("\nVehicle Service Center Management\n");
        printf("1. Add Vehicle\n");
        printf("2. Update Service History\n");
        printf("3. Display Vehicles\n");
        printf("4. Exit\n");
        printf("Enter your choice: ");
        scanf("%d", &choice);

        if (choice == 1) {
            if (vehicle_count >= 5) {
                printf("Cannot add more vehicles. Maximum capacity reached.\n");
```

```
        // continue;
    }

    printf("Enter license plate: ");
    scanf("%s", vehicles[vehicle_count].license_plate);
    printf("Enter owner's name: ");
    scanf(" %s", vehicles[vehicle_count].owner_name);
    printf("Enter vehicle type : ");
    scanf("%s", vehicles[vehicle_count].vehicle_type);



    for (int i = 0; i < 5; i++) {
        strcpy(vehicles[vehicle_count].service_history[i].service_type, "");
        vehicles[vehicle_count].service_history[i].cost = 0.0;
        strcpy(vehicles[vehicle_count].service_history[i].service_date, "");
    }

    printf("Vehicle added successfully!\n");
    vehicle_count++;

} else if (choice == 2) {
    char license_plate[15];
    printf("Enter license plate of the vehicle to update: ");
    scanf("%s", license_plate);

    int found = -1;
    for (int i = 0; i < vehicle_count; i++) {
        if (strcmp(vehicles[i].license_plate, license_plate) == 0) {
            found = i;
            break;
        }
```

```
    }

    if (found == -1) {
        printf("Vehicle with license plate %s not found.\n", license_plate);
        continue;
    }

    int service_index = -1;
    for (int j = 0; j < 5; j++) {
        if (strlen(vehicles[found].service_history[j].service_type) == 0) {
            service_index = j;
            break;
        }
    }

    if (service_index == -1) {
        printf("No space to add more services for this vehicle.\n");
        continue;
    }

    printf("Enter service type: ");
    scanf(" %s", vehicles[found].service_history[service_index].service_type);
    printf("Enter service cost: ");
    scanf("%f", &vehicles[found].service_history[service_index].cost);
    printf("Enter service date (DD/MM/YYYY): ");
    scanf("%s", vehicles[found].service_history[service_index].service_date);

    printf("Service record added successfully!\n");

} else if (choice == 3) {
    if (vehicle_count == 0) {
```

```
            printf("No vehicles to display.\n");
            continue;
        }

        for (int i = 0; i < vehicle_count; i++) {
            printf("\nVehicle %d\n", i + 1);
            printf("License Plate: %s\n", vehicles[i].license_plate);
            printf("Owner's Name: %s\n", vehicles[i].owner_name);
            printf("Vehicle Type: %s\n", vehicles[i].vehicle_type);

            printf("Service History:\n");
            for (int j = 0; j < 5; j++) {
                if (strlen(vehicles[i].service_history[j].service_type) > 0) {
                    printf(" Service %d\n", j + 1);
                    printf(" Type: %s\n", vehicles[i].service_history[j].service_type);
                    printf(" Cost: %.2f\n", vehicles[i].service_history[j].cost);
                    printf(" Date: %s\n", vehicles[i].service_history[j].service_date);
                }
            }
        }

    } else if (choice == 4) {
        printf("Exiting\n");
        break;

    } else {
        printf("Invalid choice\n");
    }
}

return 0;
```

```
}
```

Vehicle Service Center Management

1. Add Vehicle

2. Update Service History

3. Display Vehicles

4. Exit

Enter your choice: 3

Vehicle 1

License Plate: kl35j8505

Owner's Name: rin

Vehicle Type: car

Service History:

Service 1

Type: wash

Cost: 421.00

Date: 21-11-24

Vehicle Service Center Management

1. Add Vehicle

2. Update Service History

3. Display Vehicles

4. Exit

Enter your choice: 1

Enter license plate: kl35j8504

Enter owner's name: ty

Enter vehicle type : car

Vehicle added successfully!

Vehicle Service Center Management

1. Add Vehicle

2. Update Service History

3. Display Vehicles

4. Exit

Enter your choice: 2

Enter license plate of the vehicle to update: kl35j8504

Enter service type: wash

Enter service cost: 230

Enter service date (DD/MM/YYYY): 21-11-24

Service record added successfully!

Vehicle Service Center Management

1. Add Vehicle

2. Update Service History

3. Display Vehicles

4. Exit

Enter your choice: 3

Vehicle 1

License Plate: kl35j8505

Owner's Name: rin

Vehicle Type: car

Service History:

Service 1

Type: wash

Cost: 421.00

Date: 21-11-24

Vehicle 2

License Plate: kl35j8504

Owner's Name: ty

Vehicle Type: car

```
Service History:
 Service 1
 Type: wash
 Cost: 230.00
 Date: 21-11-24

Vehicle Service Center Management
1. Add Vehicle
2. Update Service History
3. Display Vehicles
4. Exit
Enter your choice: 4
Exiting the program. Goodbye!
PS D:\learning c\output>
```