PHASE 1 DAY 13

STRUCTURES AND POINTERS

1.
```
#include <stdio.h>

struct intPtrs
{
   int *p1;
   int *p2;
};

int main()

{
   struct intPtrs pointers;
   int i1=100,i2;

   pointers.p1=&i1;
   pointers.p2=&i2;
   *pointers.p2=180;

   printf("i1 = %d *pointers.p1= %d\n",i1,*pointers.p1);
   printf("i2 = %d *pointers.p2= %d\n",i2,*pointers.p2);

   return 0;
}
```
Output:
```
i1 = 100 *pointers.p1= 100
i2 = 180 *pointers.p2= 180
```

2.
```
#include <stdio.h>

struct names{

   char first[40];

   char last[40];

};

struct pNames{
```

```
    char *first;
    char *last;
};
int main(){
    struct names CAnames ={"Abhinav", "Karan"};
    struct pNames CPnames = {"Abhinav","Karan"};

    printf("%s\t%s \n",CAnames.first,CPnames.first);
    printf("size of CAnames = %d\n",sizeof(CAnames));
    printf("size of CPnames = %d\n",sizeof(CPnames));
    return 0;
}
```

STRUCTURE AND FUNCTIONS

1.

```
#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 CAnames ={"Abhinav", "Karan"};
	struct names CPnames = {"Abhinav","Karan"};

	bool b = nameCOmparison(CAnames, CPnames);

	printf("b = %d",b);

	return 0;
}

bool nameCOmparison(struct names CAnames, struct names CPnames){
	if(strcmp(CAnames.first,CPnames.first) == 0){
	return true;
	}
	else{
	return false;
	}
}
```

2.

```
#include <stdio.h>
 #include <string.h>
 #include <stdbool.h>
 #include <time.h>
 struct names{
	char first[40];
```

```
        char last[40];
};


bool nameCOmparison(struct names *, struct names *);


int main(){
        clock_t start, end;
        double cpu_used_time;

        start = clock();

        struct names CAnames ={"Abhinav", "Karan"};
        struct names CPnames = {"Abhinav","Karan"};

        //struct names *ptr1, *ptr2;

        //ptr1 = &CAnames;
        //ptr2 = &CPnames;


        bool b = nameCOmparison(&CAnames, &CPnames);

        printf("b = %d",b);

        end = clock();

        cpu_used_time = ((double)(end - start)) / CLOCKS_PER_SEC;

        printf("cpu_used_time = %f \n",cpu_used_time);


        return 0;
}


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

```
        }
}
```

Problem 1: Dynamic Student Record Management

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

Description:

Define a structure Student with fields:

1.
    - **int roll_no: Roll number**
    - **char *name: Pointer to dynamically allocated memory for the student's name**
    - **float marks: Marks obtained**

Write a program to:

2.
    - **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.**
    -

Answer:

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

struct Student
{
   int roll_no;
   char *name;
   float marks;
};

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

   struct Student *stud = (struct Student *)malloc(n * sizeof(struct Student));
  if (stud == NULL)
```

```
  {
      printf("Memory allocation failed.\n");
      return 1;
  }

  for (int i = 0; i < n; i++)
  {
      printf("\nEnter details for student %d:\n", i + 1);
      printf("Roll number: ");
      scanf("%d", &stud[i].roll_no);

      stud[i].name = (char *)malloc(100 * sizeof(char));

      if (stud[i].name == NULL)
      {
          printf("Memory allocation failed for name.\n");
          return 1;
      }

      printf("Name: ");
      scanf(" %[^\n]", stud[i].name);
      getchar();
      printf("Marks: ");
      scanf("%f", &stud[i].marks);
  }

  printf("\nStudent Details:\n");
  for (int i = 0; i < n; i++)
  {
      printf("Roll No: %d\nName: %s\nMarks: %.2f\n", stud[i].roll_no, stud[i].name, stud[i].marks);
  }

  for (int i = 0; i < n; i++)
  {
      free(stud[i].name);
  }

  free(stud);

  return 0;
}
```

Problem 2: Library System with Dynamic Allocation

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

Description:

Define a structure Book with fields:

1.
   - **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)**

Write a program to:

2.
   - **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.**

Answer:

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

struct Book
{
   char *title;
   char *author;
   int *copies;
};

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

   struct Book *books = (struct Book *)malloc(n * sizeof(struct Book));

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

   for (int i = 0; i < n; i++)
   {
      printf("\nEnter details for book %d:\n", i + 1);
```

```
    books[i].title = (char *)malloc(100 * sizeof(char));

    if (books[i].title == NULL)
    {
        printf("Memory allocation failed for title.\n");
        return 1;
    }
    printf("Title: ");
    scanf(" %[^\n]", books[i].title);
    getchar();

    books[i].author = (char *)malloc(100 * sizeof(char));
    if (books[i].author == NULL)
    {
        printf("Memory allocation failed for author.\n");
        return 1;
    }
    printf("Author: ");
    scanf(" %[^\n]", books[i].author);
    getchar();

    books[i].copies = (int *)malloc(sizeof(int));

    if (books[i].copies == NULL)
    {
        printf("Memory allocation failed for copies.\n");
        return 1;
    }

    printf("Number of copies: ");
    scanf("%d", books[i].copies);
}

 printf("\nBook Details:\n");
for (int i = 0; i < n; i++)
{
    printf("Title: %s\nAuthor: %s\nCopies: %d\n", books[i].title, books[i].author, *books[i].copies);
}

int bookindex, newcopies;

printf("\nEnter the index of the book to update (1 to %d): ", n);
scanf("%d", &bookindex);

if (bookindex < 1 || bookindex > n)
{
```

```
        printf("Invalid index.\n");
    }
    else
    {
        printf("Enter the new number of copies: ");
        scanf("%d", &newcopies);

        *books[bookindex - 1].copies = newcopies;
        printf("Updated Details: Title: %s, Author: %s, Copies: %d\n",
               books[bookindex - 1].title,
               books[bookindex - 1].author,
               *books[bookindex - 1].copies);
    }

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

    return 0;
}
```

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.**

Answer:

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

```
struct Complex
{
    float real;
    float imag;
};


struct Complex addComplex(struct Complex c1, struct Complex c2)
{
    struct Complex result;
    result.real = c1.real + c2.real;
    result.imag = c1.imag + c2.imag;
    return result;
}


struct Complex multiplyComplex(struct Complex c1, struct Complex c2)
{
    struct Complex result;
    result.real = (c1.real * c2.real) - (c1.imag * c2.imag);
    result.imag = (c1.real * c2.imag) + (c1.imag * c2.real);
    return result;
}

int main()
```

```
{
    struct Complex c1, c2, sum, product;

    printf("Enter the real and imaginary parts of the first complex number: ");
    scanf("%f %f", &c1.real, &c1.imag);

    printf("Enter the real and imaginary parts of the second complex number: ");
    scanf("%f %f", &c2.real, &c2.imag);

    sum = addComplex(c1, c2);
    product = multiplyComplex(c1, c2);

    printf("\nThe sum of the complex numbers is: %.2f + %.2fi\n", sum.real, sum.imag);
    printf("The product of the complex numbers is: %.2f + %.2fi\n", product.real, product.imag);

    return 0;
}
```

```
Enter the real and imaginary parts of the first complex number: 2
3
Enter the real and imaginary parts of the second complex number: 2
6

The sum of the complex numbers is: 4.00 + 9.00i
The product of the complex numbers is: -14.00 + 18.00i
```

Problem 2: Rectangle Area and Perimeter Calculator

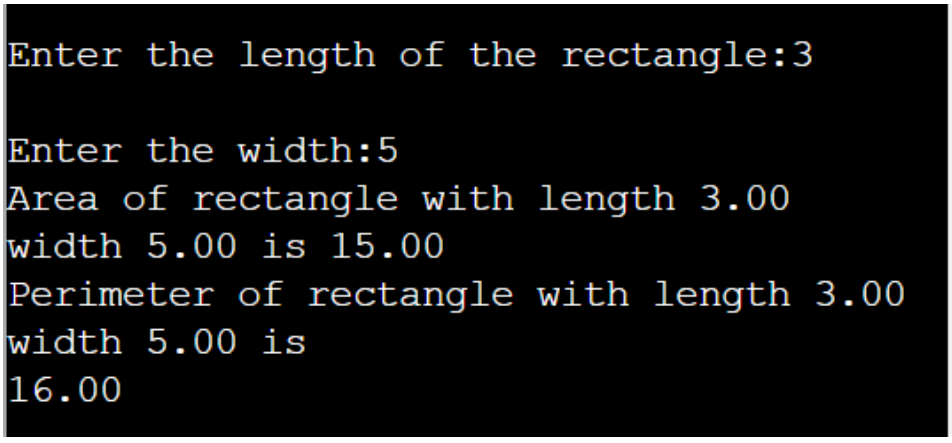
Objective: Calculate the area and perimeter of a rectangle by passing a structure to functions.

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.**

Answer:

```
#include<stdio.h>

#include<string.h>

#include<stdlib.h>



struct rectangle

{

  float length;

  float width;

};

float area_rectangle(struct rectangle);

float perimeter_rectangle(struct rectangle);


int main()

{

    struct rectangle rect;

    printf("\nEnter the length of the rectangle:");
```



```
    scanf("%f",&rect.length);


    printf("\nEnter the width:");
    scanf("%f",&rect.width);


    float area =area_rectangle(rect);
    printf("Area of rectangle with length %.2f\nwidth %.2f is %.2f",rect.length,rect.width,area);


    float perimeter=perimeter_rectangle(rect);
    printf("\nPerimeter of rectangle with length %.2f\nwidth %.2f
is\n%.2f",rect.length,rect.width,perimeter);


    return 0;
}


float area_rectangle(struct rectangle rect)
{
    return rect.length*rect.width;
}


float perimeter_rectangle(struct rectangle rect)
{
    return 2*(rect.length+rect.width);
}
```

Problem 3: Student Grade Calculation

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

Description:

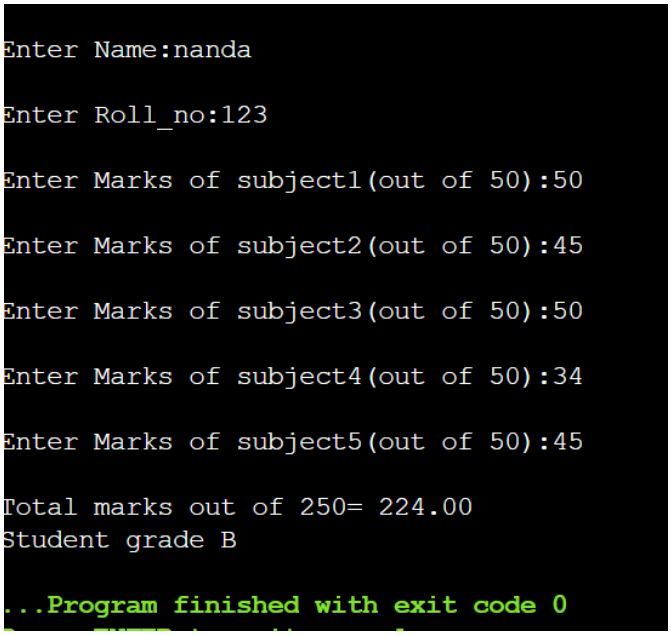
1. **Define a structure Student with fields:**
    - **char name[50]: Name of the student**
    - **int roll_no: Roll number**
    - **float marks[5]: Marks in 5 subjects**
    - **char grade: Grade assigned to the student**
2. **Write a function to:**
    - **Calculate the average marks and assign a grade (A, B, etc.) based on predefined criteria.**
3. **Pass the structure by reference to the function and modify the grade field.**

Answer:

```
#include<stdio.h>

#include<string.h>

#include<stdlib.h>



struct Student

{

    char name[50];

    int roll_no;

    float marks[5];

    char grade;

};

float Grade(struct Student*);



int main()

{

    struct Student stud;
```

```
    printf("\nEnter Name:");
    scanf("%[^\n]",stud.name);
    getchar();

    printf("\nEnter Roll_no:");
    scanf("%d",&stud.roll_no);

   for(int i=0;i<5;i++)
   {
    printf("\nEnter Marks of subject%d(out of 50):",i+1);
    scanf("%f",&stud.marks[i]);
   }
   stud.grade=Grade(&stud);
    printf("\nStudent grade %c",stud.grade);

    return 0;
}

float Grade(struct Student * stud_ent)
{
    float sum=0;

    for(int i=0;i<5;i++)
```

```
    {
        sum += stud_ent->marks[i];
    }

    printf("\nTotal marks out of 250= %.2f",sum);

    if(sum>=225)
    {
        stud_ent->grade='A';
    }
    else if(sum>=175 && sum<225)
    {
        stud_ent->grade='B';
    }
     else if(sum>=125 && sum<175)
    {
        stud_ent->grade='C';
    }
    else
    {
       stud_ent->grade='D';
    }

      return stud_ent->grade;
}
```



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.**

Answer:

```
#include<stdio.h>

#include<math.h>

#include<stdlib.h>



struct Point

{

   float x;

   float y;

}p;



float distance_btw_points(struct Point,struct Point);

void lies_inside_circle(struct Point,float radius);



int main()
```

```
{
    struct Point p1;
    struct Point p2;
    float radius;
    printf("Enter the coordinates of point 1\nX:");
    scanf("%f",&p1.x);

    printf("\nY:");
    scanf("%f",&p1.y);

   printf("Enter the coordinates of point 2\nX:");
    scanf("%f",&p2.x);

    printf("\nY:");
    scanf("%f",&p2.y);

    float distance=distance_btw_points(p1,p2);
    printf("\nDistance between(%.1f,%.1f) and (%.1f,%.1f) is %.1f",p1.x,p1.y,p2.x,p2.y,distance);

    printf("\nEnter radius:");
    scanf("%f",&radius);

    lies_inside_circle(p, radius);
    lies_inside_circle(p, radius);
```

```
    return 0;
}

float distance_btw_points(struct Point pt1,struct Point pt2)
{
    return sqrt(pow(pt2.x - pt1.x, 2) + pow(pt2.y - pt1.y, 2));

}

void lies_inside_circle(struct Point pt, float radius)

{
    printf("Enter the coordinates of point \nX:");
    scanf("%f",&pt.x);

    printf("\nY:");
    scanf("%f",&pt.y);

    float distance_from_origin = sqrt(pow(pt.x, 2) + pow(pt.y, 2));

    if (distance_from_origin <= radius)
    {
        printf("Point (%.1f, %.1f) lies inside the circle.\n", pt.x, pt.y);
    }
    else
```

```
    {
        printf("Point (%.1f, %.1f) does not lie inside the circle.\n", pt.x, pt.y);
    }
}
```

Problem 5: Employee Tax Calculation

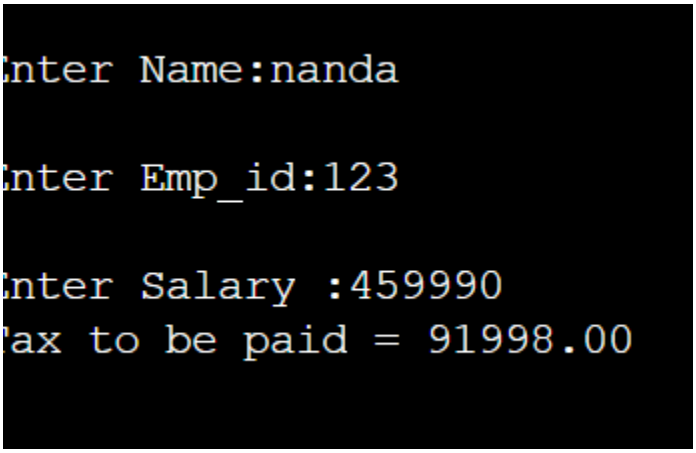
Objective: Calculate income tax for an employee based on their salary by passing a structure to a function.

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.**

Answer:

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

struct Employee
{
   char name[50];
   int emp_id;
   float salary;
   float tax;
};
float Tax_Salary(struct Employee*);

int main()
{
   struct Employee Emp;
   Emp.tax=0;
```



```
    printf("\nEnter Name:");
    scanf("%[^\n]",Emp.name);
    getchar();

    printf("\nEnter Emp_id:");
    scanf("%d",&Emp.emp_id);

    printf("\nEnter Salary :");
    scanf("%f",&Emp.salary);

    Emp.tax=Tax_Salary(&Emp);
    printf("Tax to be paid = %.2f\n",Emp.tax);


    return 0;
}

float Tax_Salary(struct Employee *Emp)
{
    if(Emp->salary>=50000)
    {
        Emp->tax = 0.2*Emp->salary;
    }
    else
    {
        Emp->tax=0.1*Emp->salary;
    }

    return  Emp->tax;
}
```

Problem Statement: Vehicle Service Center Management

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

Description:

Define a structure Vehicle with fields:

1.
    - **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)**

Define a nested structure Service inside Vehicle with fields:

2.
    - **char service_type[30]: Type of service performed**
    - **float cost: Cost of the service**
    - **char service_date[12]: Date of service**

Implement the following features:

3.
    - **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.**

Answer:

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

#define MAX 100


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 services[MAX];
    int service_count;
};

struct Vehicle vehicles[MAX];
int vehicle_count = 0;

void addVehicle();
void updateService();
void displayServiceDetails();
void generateSummaryReport();

int main() {
    int choice;
```

```
    while (1) {
        printf("\nVehicle Service Center Management\n");
        printf("1. Add Vehicle\n");
        printf("2. Update Service History\n");
        printf("3. Display Service Details\n");
        printf("4. Generate Summary Report\n");
        printf("5. Exit\n");
        printf("Enter your choice: ");
        scanf("%d", &choice);
        getchar();

        switch (choice) {
            case 1:
                addVehicle();
                break;
            case 2:
                updateService();
                break;
            case 3:
                displayServiceDetails();
                break;
            case 4:
                generateSummaryReport();
                break;
            case 5:
                return 0;
            default:
                printf("Invalid\n");
        }
    }

    return 0;
}

void addVehicle()
{
    if (vehicle_count >= MAX)
    {
        printf("Vehicle storage is full. Cannot add more vehicles.\n");
        return;
    }

    printf("Enter license plate: ");
    scanf(" %[^\n]", vehicles[vehicle_count].license_plate);
    getchar();
```

```
    printf("Enter owner's name: ");
    scanf(" %[^\n]", vehicles[vehicle_count].owner_name);
    getchar();

    printf("Enter vehicle type (e.g., car, bike): ");
    scanf(" %[^\n]", vehicles[vehicle_count].vehicle_type);
    getchar();

    vehicles[vehicle_count].service_count = 0;
    vehicle_count++;

    printf("Vehicle added successfully!\n");
}

void updateService()
{
    if (vehicle_count == 0)
    {
        printf("No vehicles added yet.\n");
        return;
    }

    char license_plate[15];
    printf("Enter license plate of the vehicle to update service: ");
    scanf(" %[^\n]", license_plate);
    getchar();

    for (int i = 0; i < vehicle_count; i++) {
        if (strcmp(vehicles[i].license_plate, license_plate) == 0)
        {
            if (vehicles[i].service_count >= MAX)
            {
                printf("Service record limit reached for this vehicle.\n");
                return;
            }

            struct Service *new_service = &vehicles[i].services[vehicles[i].service_count];
            printf("Enter service type: ");
            scanf(" %[^\n]", new_service->service_type);
            getchar();

            printf("Enter service cost: ");
            scanf("%f", &new_service->cost);
            getchar();

            printf("Enter service date (dd/mm/yyyy): ");
```

```
            scanf(" %[^\n]", new_service->service_date);
            getchar();

            vehicles[i].service_count++;
            printf("Service record updated successfully!\n");
            return;
        }
    }

    printf("Vehicle with license plate %s not found.\n", license_plate);
}

void displayServiceDetails()
{
    if (vehicle_count == 0)
    {
        printf("No vehicles added yet.\n");
        return;
    }

    char license_plate[15];
    printf("Enter license plate of the vehicle: ");
    scanf(" %[^\n]", license_plate);
    getchar();

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

            if (vehicles[i].service_count == 0)
            {
                printf("No service records found for this vehicle.\n");
                return;
            }

            printf("Service Records:\n");
            for (int j = 0; j < vehicles[i].service_count; j++)
            {
                struct Service *svc = &vehicles[i].services[j];
                printf("  Service %d:\n", j + 1);
                printf("    Type: %s\n", svc->service_type);
                printf("    Cost: %.2f\n", svc->cost);
```

```
            printf("    Date: %s\n", svc->service_date);
        }
        return;
    }
  }

  printf("Vehicle with license plate %s not found.\n", license_plate);
}

void generateSummaryReport()
{
  if (vehicle_count == 0)
  {
    printf("No vehicles added yet.\n");
    return;
  }

  float total_revenue = 0;
  printf("Summary Report:\n");
  printf("License Plate\tOwner Name\tTotal Cost\n");

  for (int i = 0; i < vehicle_count; i++)
  {
    float vehicle_total = 0;
    for (int j = 0; j < vehicles[i].service_count; j++)
    {
      vehicle_total += vehicles[i].services[j].cost;
    }
    total_revenue += vehicle_total;

    printf("%s\t%s\t%.2f\n",
        vehicles[i].license_plate,
        vehicles[i].owner_name,
        vehicle_total);
  }

  printf("Total Revenue: %.2f\n", total_revenue);
}
```