1. Create structure item details with members itemname, quantity, price, total amount. Calculate party expenses.

⇒

```
#include <stdio.h>
#include <string.h>
struct Item {
    char item_name[50];
    int quantity;
    float price;
    float total_amount;
};
float calculate_total_amount(int quantity,
  float prize) { return quantity * price;
}
int main()
{
  struct Item item1 = {"Item1", 3, 10.0, 0.0};
  struct Item item2 : {"Item2", 2, 15.0, 0.0};
  struct Item item3 : {"Item3", 5, 8.0, 0.0};
item1.total_amount = calculate_total_amount(
                    item1.quantity, item1.price);
item2.total_amount = calculate_total_amount(
                    item2.quantity, item2.price);
item3.total_amount = calculate_total_amount(
                    item3.quantity, item3.price);
  float party_expenses = item1.total_amount +
item2.total_amount + item3.total_amount;
```

```
printf("Total Party Expenses: $%.2f\n", party_expenses);
return 0;
}
```

Output:

Total Party Expenses: $155.00

---

2. Create a structure with name student with structure members: name, usn, gradelist of sem1, grade list of sem2. The student will be promoted to 3rd semester. If he/she is not having backlogs of credit count >= 16.

⇒

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

struct GradeList {
    char grades[n];
};
struct Student {
    char name[50];
    char usn[15];
    struct GradeList sem1;
    struct GradeList sem2;
};
int calculateCreditCount(struct GradeList grades) {
    int creditCount = 0;
    for (int i=0; i<n; i++) {
    if (grades.grades[i] == 'A') {
        creditCount += 4; }
    else if (grades.grades[i] == 'B') {
```

```
        creditCount += 3; }
    else if (grades.grades[i] == 'C') {
        creditCount += 2;
    }
    else if (grades.grades[i] == 'D') {
        creditCount += 1;
    }
  }
  return creditCount;
}

int main() {
  struct Student student = {
      .name = "John Doe",
      .usn = "1ABCD123",
      .sem1 = {.grades = {'A', 'B', 'C', 'A', 'B', 'A'}},
      .sem2 = {.grades = {'A', 'A', 'B', 'B', 'C', 'D'}}
  };
  int sem1Credits = calculateCreditCount(student.sem1);
  int sem2Credits = calculateCreditCount(student.sem2);
  if (sem1Credits >= 16 && sem2Credits >= 16) {
      printf("Promoted to 3rd Semester");
  } else {
      printf("Not Promoted to 3rd Semester");
  }
  return 0;
}
```

3. Create a structure with the name Student book with members: name, usn, book_id, issue_date. A student at max will be issued 4 books for 90 days. If the student crosses 90 days to return back, each day the fine for each book is 50 along with the book cost. Display the status of Student as follows.

* no of days remaining return for the books.
* no of books still student can be granted out of 4.
* any pending return with fine calculation.

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

Struct Date{int day, month, year;};
Struct Book{char book_id[20]; struct Date issue_date;};
Struct Studentbook{char name[50], usn[15]; struct
        Book books_issued[4]; int num_books_issued;};

int daysPassed(struct Date issue_date){
    Struct tm issued_date={.tm_year=issue.date.year
    -1900,.tm_mon=issue_date.month-1,.tm_m
    day=,issue_date.day};
    return difftime(time(NULL), mktime(&issued_date)
(60*60*24);
}
```

```
int main() {
struct StudentBook student = {.name = "Alice", .usn =
"1ABC123", .num_books_issued = 2,
  .books_issued = { {.book_id = "B001", .issue_date =
{1, 9, 2023}}, {.book_id = "B002", .issue_date =
{15, 9, 2023}} }
};
const int MAX_DAYS = 90, FINE_PER_DAY = 50;
int remaining_days = MAX_DAYS, - daysPassed(student.
  books_issued[0].issue_date);
int books_available = 4 - student.num_books_issued;
printf("Days remaining for book return : %d \n
Number of books available for issue : %d \n", remaining_
days, books_available);
for (int i = 0; i < student.num_books_issued; ++i) {
  int days_overdue = daysPassed (student.books_
issued[i].issue_date) - MAX_DAYS;
  if (days_overdue > 0)
    printf ("Book ID : %s, Overdue days : %d, Fine :
Rs. %d \n", student.books_issued[i].book_id,
days_overdue * FINE_PER_DAY);
}
return 0;
}
```

Output :

Days remaining for book return : 82

Number of books available for issue : 2

Book ID : B001, Overdue days : 10, Fine : Rs. 500

Book ID : B002, Overdue days : 4, Fine : Rs. 200.

Q4

=

~~#include<stdio.h>~~

~~struct distance~~

4. Given an array() containing N distances of the inch-feet system, such that each element of the array represents a distance in the form of {inch, feet}. The task is to add all the N inch-feet distances using structure.

I/p = array() = {{10, 3.7}, {10, 5.5}, {6, 8.0}};

O/p = Feet Sum: 27 Inch Sum = 5.20.

⟹

```
#include<stdio.h>
Struct distances{
  float arr[10][10];
}dist;
void main()
{  int a, i, j, FeetSum=0, InchSum=0;
```

```
printf("Enter the number of inch-feet pairs
 to be added");
scanf("%d",&a);
print+("Enter the pairs:");
for(i=0; i<a ; i++)
{  for(j=0 ; j<2 ; j++){
    Scanf("%f", &dist.arr[i][j]);
  }
}
for(i=0; i<a ; i++) {
      FeetSum+= dist.arr[i][0];
      InchSum+=dist.arr[i][1];
 }
 printf("total feet sum = %d\n", FeetSum);
 printf("total inchsum = %d\n", InchSum);
}.
```

Output:

Enter the number of Inch-feet pairs.

to be added: 3

Enter the pairs: 10, 3.7
10 5.5
6 8.0

feetSum = 26.

InchSum = 17.2