Requirements
1. Define Data Types
Product Structure:
Use a structure Fertilizer to represent details of fertilizer products with the following fields:
productID (integer): Unique identifier for the fertilizer product.
productName (string): Name of the fertilizer product.
quantityAvailable (integer): Current stock quantity of the product.
pricePerUnit (float): Price per unit of the product.
totalSales (float): Total sales value (calculated as price per unit × quantity sold).
Sales Structure:
Use a structure SalesRecord to represent sales data:
salesID (integer): Unique identifier for the sales transaction.
productID (integer): Product ID of the fertilizer sold.
quantitySold (integer): Quantity sold in the transaction.
Union for Supplier Details:
Use a union SupplierContact to store either:
phoneNumber (string): Contact number of the supplier.
email (string): Email address of the supplier.

2. Features
Dynamic Memory Allocation:
Dynamically allocate memory for:
An array of Fertilizer structures to represent all products.
An array of SalesRecord structures to track sales transactions.
Input and Output:
Input the details of each fertilizer product.
Record sales transactions, updating product stock and calculating total sales.
Input supplier contact details using the union SupplierContact.
Display:
Display details of all fertilizer products, including stock and total sales value.
Display all sales records.
Search:
Search for a product by its ID and display its details.
Search for sales transactions by salesID.
Update:
Update the stock or price of a product and recalculate its total sales if required.
Sorting:
Sort fertilizers by total sales in descending order.
Sort sales records by salesID in ascending order.

3. Typedef
Use typedef to define aliases for Fertilizer, SalesRecord, and SupplierContact.

Program Requirements
1. Menu Options
Input Fertilizer Details.
Input Sales Transaction.
Update Fertilizer Stock or Price.
Display All Fertilizer Details.
Display All Sales Records.
Search Fertilizer by Product ID.
Search Sales Record by Sales ID.
Sort Fertilizers by Total Sales.
Exit.
has context menu

```
#include<stdio.h>
#include<stdlib.h>
typedef struct{
int productID ;
char productName [50];
int quantityAvailable;
float pricePerUnit ;
float totalSales;
}Fertilizer;

typedef struct{
int salesID ;
int productID ;
int quantitySold ;

}SalesRecord;

typedef union{
  char phoneNumber [10];
  char email [30];

}SupplierContact;
void addfertilizer(Fertilizer*fertilize,SupplierContact *sup,int n);
void addsales(SalesRecord*sales,int n);
void displaysales(SalesRecord*sales,int n);
void displayfertilizer(Fertilizer*fertilize,SupplierContact *sup,int n);
void search(Fertilizer*fertilize,int n);
void searchsales(SalesRecord*sales,int n);
void decenting(Fertilizer*fertilize,int n);
void asenting(SalesRecord*sales,int n);
void updateFertilizer(Fertilizer* fertilize, int n);



int main(){
    int count=0,n,op;
    printf("enter the number of products");
    scanf("%d",&n);
    Fertilizer*fertilize = (Fertilizer*)malloc(n*(sizeof(Fertilizer)));
    SalesRecord*sales=(SalesRecord*)malloc(n*sizeof(SalesRecord));
    SupplierContact*sup=(SupplierContact*)malloc(n*sizeof(SupplierContact));
   while(1){
    printf("Input Fertilizer Details.\n Input Sales Transaction.\n Update Fertilizer Stock or Price.\n
Display All Fertilizer Details.\n Display All Sales Records.\n Search Fertilizer by Product ID.\n
Search Sales Record by Sales ID.\n Sort Fertilizers by Total Sales.\n Exit\nenter your choice");
    scanf("%d",&op);
    switch(op){
        case 1:
        addfertilizer(fertilize,sup,n);
        break;
        case 2:
        addsales(sales,n);
        break;
        case 3:
        displayfertilizer(fertilize,sup,n);
        break;
        case 4:
        displaysales(sales,n);
        break;
        case 5:
        search(fertilize,n);
        break;
        case 6:
        searchsales(sales,n);
        break;
        case 7:
        decenting(fertilize,n);
        break;
        case 8:
```

```
        asenting(sales,n);
        break;
        case 9:
         updateFertilizer(fertilize,n);
        break;


    }







   }


}
void addfertilizer(Fertilizer*fertilize,SupplierContact *sup,int n){
    for(int i=0;i<n;i++){
        printf("enter the product id");
        scanf("%d",&fertilize[i].productID);
        printf("enter the name");
        scanf("%s",fertilize[i].productName);
        printf("enter the quality");
        scanf("%d",&fertilize[i].quantityAvailable);
        printf("enter the price");
        scanf("%f",&fertilize[i].pricePerUnit);
        fertilize[i].totalSales=fertilize[i].pricePerUnit*fertilize[i].quantityAvailable;
        int m;
        printf("enter 1 for phone number id 2 for email id");
        scanf("%d",&m);
        if(m==1){
            printf("enter the phone number");
            scanf("%s",sup[i].phoneNumber);
        }else{
             printf("enter the email id");
            scanf("%s",sup[i].email);

        }
    }
}
void addsales(SalesRecord*sales,int n){
    for(int i=0;i<n;i++){
        printf("enter the product id");
        scanf("%d",&sales[i].productID);
        printf("enter the name");
        scanf("%d",&sales[i].salesID);
        printf("enter the quality");
        scanf("%d",&sales[i].quantitySold);

    }
}
void displaysales(SalesRecord*sales,int n){
    for(int i=0;i<n;i++){
       printf("sales id:%d \n product id:%d \n quality id:%d\n
",sales[i].productID,sales[i].productID,sales[i].quantitySold);

    }
}
void displayfertilizer(Fertilizer*fertilize,SupplierContact *sup,int n){
    for(int i=0;i<n;i++){
      printf("product id:%d\nname:%s\nprice:%f\nquality%d\nContact
Details:%s",fertilize[i].productID,fertilize[i].productName,fertilize[i].pricePerUnit,fertilize[i].quantity
Available,sup[i].phoneNumber);
    }
}
void search(Fertilizer*fertilize,int n){
    int v;
    int fount=0;
    printf("enter the id");
    printf("-----------------------");
    scanf("%d",&v);
    for(int i=0;i<n;i++){
        if(fertilize[i].productID==v){
            printf("name:%s\nproduct id:%d
\nprice:%f\nquality:%d\n",fertilize[i].productName,fertilize[i].productID,fertilize[i].pricePerUnit,ferti
lize[i].quantityAvailable);
            fount=1;
        }
    }
    if(!fount){
        printf("the id is not found in here");
    }
}
// Search Sales Record by Sales ID
void searchsales(SalesRecord*sales,int n){
    int v;
    int fount=0;
    printf("enter the id");
    printf("-----------------------");
    scanf("%d",&v);
    for(int i=0;i<n;i++){
        if(sales[i].productID==v){
            printf("salesid:%d\n sold id:%d\n
quality:%d",sales[i].productID,sales[i].salesID,sales[i].quantitySold);
            fount=1;
        }
    }
    if(!fount){
        printf("the id is not found in here");
    }
}

// sort
void decenting(Fertilizer*fertilize,int n){
    Fertilizer temp;
    for(int i=0;i<n-1;i++){
        for(int j=0;j<n-i-1;j++){
            if(fertilize[j].totalSales<fertilize[j+1].totalSales){
            temp=fertilize[j];
            fertilize[j]=fertilize[j+1];
            fertilize[j+1]=temp;
        }

        }

    }
}
// Sort sales records by salesID in ascending order.

void asenting(SalesRecord*sales,int n){
    SalesRecord temp;
    for(int i=0;i<n-1;i++){
        for(int j=0;j<n-i-1;j++){
            if(sales[j].salesID>sales[j+1].salesID){
            temp=sales[j];
            sales[j]=sales[j+1];
            sales[j+1]=temp;
        }

        }
    }
}

void updateFertilizer(Fertilizer* fertilize, int n) {
    int id, found = 0;
    printf("Enter the product ID to update: ");
    scanf("%d", &id);

    for (int i = 0; i < n; i++) {
        if (fertilize[i].productID == id) {
            found = 1;
            printf("Enter new quantity current: %d: ", fertilize[i].quantityAvailable);
            scanf("%d", &fertilize[i].quantityAvailable);
            printf("Enter new price per unit current: %.2f: ", fertilize[i].pricePerUnit);
            scanf("%f", &fertilize[i].pricePerUnit);
            fertilize[i].totalSales = fertilize[i].pricePerUnit * fertilize[i].quantityAvailable;
            printf("Fertilizer updated successfully.\n");
            break;
        }
    }
    if (!found) {
        printf("Product ID not found.\n");
    }
}
```