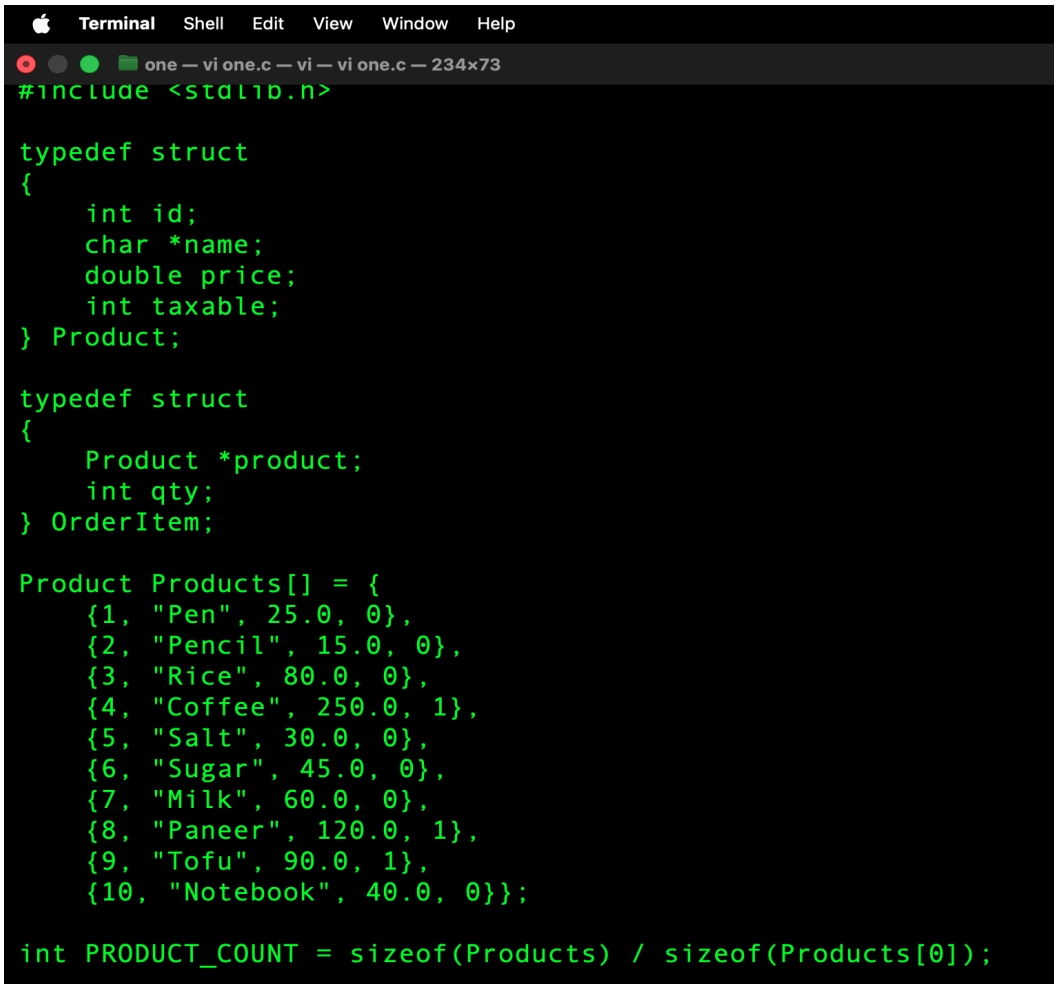
Pratik Dhimal
2407779
TASK_ONE

The Entire code is in this GIST : *CODE*

One.c

In this task these are the thing that I have done:

1. Declared a type out of structure **Product** and **OrderItem**
2. Initialized an array of **Products.**
3. Calculated the size of the **Products** array.



4. Created a function to **return the memory address** of the product with the given id and another function to just **print the product** list.

```
// If the product is not found, it returns NULL
Product *find_product_by_id(int id)

{
    for (int i = 0; i < PRODUCT_COUNT; i++)
    {
        if (Products[i].id == id)
            return &Products[i];
    }
    return NULL;
}

void print_products_list()
{
    printf("┌────┬────────────┬─────────┬─────────┐\n");
    printf("| ID | Name       | Price   | Taxable |\n");
    printf("├────┼────────────┼─────────┼─────────┤\n");
    for (int i = 0; i < PRODUCT_COUNT; i++)
    {
        printf("| %2d | %-10s | %6.2f |    %s   |\n",
               Products[i].id,
               Products[i].name,
               Products[i].price,
               Products[i].taxable ? "Yes" : "No");
    }
    printf("└────┴────────────┴─────────┴─────────┘\n");
}
```

5. Here in the main function I have allocated memory for 2 productsItem. Then later based on the cart item and quantity the cart can hold I have updated and increased the memory in run time using the **realloc()** function. This makes sure **only the` required memory is consumed**
6. Also necessary input check are added to check the user given input and validate it so that no incorrect input crashes our system.

```
}

int main()
{
    OrderItem *cart = malloc(2 * sizeof(OrderItem));
    int capacity = 2;
    int item_count = 0;
    int invoice_no = 1;

    // Read the new invoice number from file
    FILE *invFile = fopen("invoiceNumber.txt", "r");
    if (invFile)
    {
        fscanf(invFile, "%d", &invoice_no);
        fclose(invFile);
    }

    print_products_list();

    while (1)
    {
        int pid;
        int qty;
        char cont;

        while (1)
        {
            printf("\nEnter Product ID: ");
            if (scanf("%d", &pid) != 1)
            {
                // Clear invalid input
                while (getchar() != '\n')
                    ;
                printf("Invalid input. Please enter a valid Product ID.\n");
                continue;
            }

            Product *p = find_product_by_id(pid);
            if (!p)
            {
                printf("Invalid Product ID. Please try again.\n");
                continue;
            }
            break;
        }

        while (1)
        {
            printf("Enter Quantity: ");
            if (scanf("%d", &qty) != 1 || qty <= 0)
            {
                // Clear invalid input
                while (getchar() != '\n')
                    ;
                printf("Invalid Quantity. Please enter a positive number.\n");
                continue;
            }
            break;
        }

        if (item_count == capacity)
        {
            int new_capacity = capacity == 0 ? 2 : capacity * 2;
            OrderItem *temp = realloc(cart, new_capacity * sizeof(OrderItem));
            if (!temp)
            {
                printf("Memory allocation failed\n");
                free(cart);
                return 1;
            }
            cart = temp;
            capacity = new_capacity;
```

7. Lastly the invoice is generated: just a simple print statement running in a loop to print all the items in the cart. Here the **invoice number** is taken from a text field where the next invoice number is stored whenever an invoice is generated so that the next bill can take that number from the file.

```
        break;
    }

    printf("\n┌──────────────────────────────────────────────┐\n");
    printf("│              INVOICE  #%04d                  │\n", invoice_no);
    printf("├──────┬──────────────┬──────────┬──────┬────────┤\n");
    printf("│ SN   │ Product      │ Quantity │ Rate │ Total  │\n");
    printf("├──────┼──────────────┼──────────┼──────┼────────┤\n");

    double grand_total = 0.0;
    for (int i = 0; i < item_count; i++)
    {
        Product *p = cart[i].product;
        int q = cart[i].qty;
        double total = p->price * q;
        grand_total += total;
        printf("│ %2d │ %-12s │ %8d │ %4.2f │ %6.2f │\n",
               i + 1, p->name, q, p->price, total);
    }

    printf("├──────┴──────────────┴──────────┴──────┴────────┤\n");
    printf("│                            Grand Total: %6.2f │\n", grand_total);
    printf("└──────────────────────────────────────────────┘\n");

    // Write the next invoice number to file
    invFile = fopen("invoiceNumber.txt", "w");
    if (invFile)
    {
        fprintf(invFile, "%d\n", invoice_no + 1);
        fclose(invFile);
    }

    free(cart);
    return 0;
}
```

8.OUTPUT

```
[➜  one git:(master) ✗ gcc one.c -o one
[➜  one git:(master) ✗ ./one
```

ID	Name	Price	Taxable
1	Pen	25.00	No
2	Pencil	15.00	No
3	Rice	80.00	No
4	Coffee	250.00	Yes
5	Salt	30.00	No
6	Sugar	45.00	No
7	Milk	60.00	No
8	Paneer	120.00	Yes
9	Tofu	90.00	Yes
10	Notebook	40.00	No

```
Enter Product ID: 1
Enter Quantity: 3
Do you want to place another order? (y/n): y

Enter Product ID: 4
Enter Quantity: 5
Do you want to place another order? (y/n): y

Enter Product ID: a
Invalid input. Please enter a valid Product ID.

Enter Product ID: 7
Enter Quantity: 26
Do you want to place another order? (y/n): n
```

INVOICE #0006

SN	Product	Quantity	Rate	Total
1	Pen	3	25.00	75.00
2	Coffee	5	250.00	1250.00
3	Milk	26	60.00	1560.00

Grand Total: 2885.00

```
[➜  one git:(master) ✗ pwd
/Users/pratikdhimal/Developer/sem_5_hpc/TASK/one
[➜  one git:(master) ✗ whoami
pratikdhimal
➜  one git:(master) ✗
```