Name : Dheeraj, Aniket, OM

Roll No. : 54,55,

Srn no: 202201649,202201659,

Guided By: Trupti Mam

Introduction:

Vending machine codes are typically unique sequences of numbers that can be entered on the vending machine's keypad to unlock hidden features, such as free products, discounts, or diagnostic tools. These codes are often used by vending machine owners and operators, but some of them may also be available to the public.

There are several different types of vending machine codes, including:

* Service codes: These codes allow vending machine owners and operators to perform various tasks, such as setting product prices, testing products, and collecting sales data.
* Diagnostic codes: These codes can be used to troubleshoot vending machine problems.
* Promotional codes: These codes may be offered to customers as a way to provide discounts or free products.
* Secret codes: These codes are not officially documented and may not work on all vending machines. They are often rumored to provide free products or other hidden features.

It is important to note that vending machine codes can vary depending on the make and model of the machine. In addition, some vending machine companies may change their codes on a regular basis to prevent unauthorized access.

Steps And Algorithum:

Algorithum:

1. Create a linked list of food items.

2. Display the menu to the user.

3. Prompt the user to enter their choice.

4. Switch on the user's choice and perform the following actions:

\* Case 1: Display the food items in the linked list.

\* Case 2: Add a food item to the cart.

\* Case 3: View the cart and checkout.

\* Case 4: Exit the program.

5. Repeat steps 3-4 until the user selects option 4 to exit the program.

6. Free the memory allocated for the linked list.

7. Terminate the program.

Steps:

Step 1: Create a linked list of food items.

This can be done by creating a struct to represent a food item, which will contain the name and price of the food item. Then, create a linked list of food items by allocating memory for each new node and adding it to the linked list.

Step 2: Display the menu to the user.

This can be done by printing a list of the menu options to the console.

Step 3: Prompt the user to enter their choice.

This can be done by using the scanf() function to read the user's input from the console.

Step 4: Switch on the user's choice and perform the following actions:

This can be done using a switch statement to check the user's choice and perform the corresponding action.

* Case 1: Display the food items in the linked list.

This can be done by traversing the linked list and printing the name and price of each food item to the console.

* Case 2: Add a food item to the cart.

This can be done by prompting the user to enter the item number of the food item they want to add. If the item number is valid, add the food item to the cart and update the total bill.

* Case 3: View the cart and checkout.

This can be done by displaying the contents of the cart and the total bill to the console.

* Case 4: Exit the program.

This can be done by freeing the memory allocated for the linked list and terminating the program.

Step 5: Repeat steps 3-4 until the user selects option 4 to exit the program.

This can be done using a while loop to keep repeating the menu loop until the user selects option 4 to exit the program.

Step 6: Free the memory allocated for the linked list.

This can be done by traversing the linked list and freeing the memory allocated for each node.

Step 7: Terminate the program.

Code:

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

typedef struct FoodItem {

    char name[50];

    float price;

    struct FoodItem\* next;

} FoodItem;

FoodItem\* head = NULL;

//singly linked list

void addFoodItem(const char\* name, float price) {

    FoodItem\* newItem = (FoodItem\*)malloc(sizeof(FoodItem));

    strcpy(newItem->name, name);

    newItem->price = price;

    newItem->next = head;

    head = newItem;

}

//singly linked list

void displayFoodItems() {

    FoodItem\* current = head;

    int itemNumber = 1;

    while (current != NULL) {

        printf("%d. %s - $%.2f\n", itemNumber, current->name, current->price);

        current = current->next;

        itemNumber++;

    }

}

int main() {

    int choice;

    float totalBill = 0;

    addFoodItem("Chocolate", 10.0);

    addFoodItem("Chips", 15.0);

    addFoodItem("Lays", 20.0);

    addFoodItem("Kukura", 25.0);

    addFoodItem("Sprite", 30.0);

    while (1) {

        printf("Menu:\n");

        printf("1. Display Food Items\n");

        printf("2. Add to Cart\n");

        printf("3. View Cart and Checkout\n");

        printf("4. Exit\n");

        printf("Enter your choice: ");

        scanf("%d", &choice);

        switch (choice) {

            case 1:

                printf("Food Items:\n");

                displayFoodItems();

                break;

            case 2:

                printf("Enter the item number (1-5) to add to your cart: ");

                int itemNumber;

                scanf("%d", &itemNumber);

                if (itemNumber >= 1 && itemNumber <= 5) {

                    FoodItem\* current = head;

                    for (int i = 1; i < itemNumber; i++) {

                        current = current->next;

                    }

                    printf("Added %s to your cart.\n", current->name);

                    totalBill += current->price;

                } else {

                    printf("Invalid item number. Please try again.\n");

                }

                break;

            case 3:

                printf("Your Cart:\n");

                printf("Total Bill: $%.2f\n", totalBill);

                break;

            case 4:

                while (head != NULL) {

                    FoodItem\* temp = head;

                    head = head->next;

                    free(temp);

                    printf("Thank You");

                }

                return 0;

            default:

                printf("Invalid choice. Please try again.\n");

        }

    }

    return 0;

}

Output:



