EEE60104 PROGRAMMING TECHNIQUES ASSIGNMENT

NAME: SURRYARAJ A/L POOBALAN

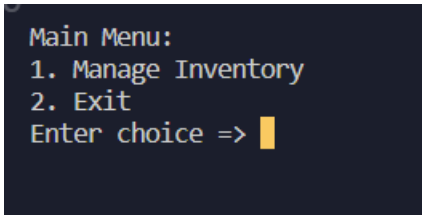
STUDENT ID: 0353127

PROGRAMME: ROBOTIC DESIGN AND DEVELOPMENT

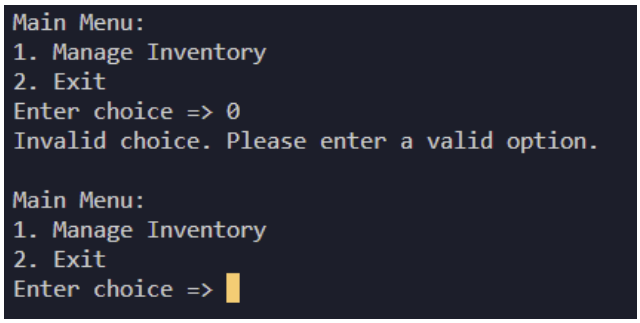
Contents

Description of Program

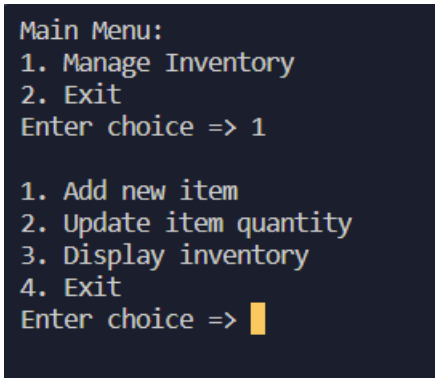
This program is an inventory management tool for grocery stores. Owners and workers of the grocery store can utilize this program to manage their stock in a simpler fashion, whether to add new items, modify quantities of existing items, or display the current stock of the items. This program aims to give grocery stores an easier way to manage their inventory by providing a more organized and up-to-date inventory, employing fewer errors and smoother operations. Below is how the program works from the user's view of point:



The program welcomes the user with a main menu display to interact with. There are two options from which the user can input to proceed with the program, '1' to access the 'Manage Inventory' and '2' to 'Exit' the program.



If the user enters an invalid option in the main menu, any option aside from 1 or 2, an alert will be displayed, "Invalid choice. Please enter a valid option.", and the user will be prompted back to the main menu.



When the user inputs 1 in the main menu, the inventory management section will be displayed. Here, the user has four inputs to choose from to continue with this section. '1' to 'Add new item', '2' to 'Update item quantity', '3' to 'Display inventory', and '4' to 'Exit'.

```
1. Add new item
2. Update item quantity
3. Display inventory
4. Exit
Enter choice => 2
Inventory is empty. Cannot update quantity.

Main Menu:
1. Manage Inventory
2. Exit
Enter choice => 3
Invalid choice. Please enter a valid option.

Main Menu:
1. Manage Inventory
2. Exit
Enter choice =>
```

When the inventory is empty, an alert will be displayed, “Inventory is empty. Cannot update quantity.”, if the user inputs ‘2’ as the choice and, “Inventory is empty.”, if ‘3’ is inputted. This will loop back the program and display the main menu.

```
1. Add new item
2. Update item quantity
3. Display inventory
4. Exit
Enter choice => 1
Enter the name of the new item => Apple
Enter the price of Apple => RM2.30
Enter the quantity of Apple => 45
Item added to the inventory.

Main Menu:
1. Manage Inventory
2. Exit
Enter choice =>
```

If ‘1’ is inputted in ‘Inventory Management’, questions will be prompted to the user to enter details regarding the new item. The user will have to enter the name, price, and quantity of the item to be added into the inventory. A successful attempt will display a success message, “Item added to the inventory.”, and bring the user back to main menu.

```
1. Add new item
2. Update item quantity
3. Display inventory
4. Exit
Enter choice => 2
Enter the name of the item => app
Item not found in the inventory.
```

When there are items in the inventory, choice ‘2’ and ‘3’ will function. Inputting ‘2’ will prompt the user to enter the name of the item. The program then searches for the item in the inventory. If the program is unable to locate the item, and alert will be displayed, “Item not found in the inventory.”.

```
1. Add new item
2. Update item quantity
3. Display inventory
4. Exit
Enter choice => 2
Enter the name of the item => Apple
Enter the new quantity for Apple => 54
Quantity updated for Apple.
```

However, if the program locates the item in the inventory, the user will further be prompted to enter the new quantity for that item to be displayed in the inventory. A successful attempt will a display success message, “Quantity for [item] updated.”.

```
1. Add new item
2. Update item quantity
3. Display inventory
4. Exit
Enter choice => 3

Current Inventory:
--------------------------------------------------
Item Name           Price(RM)      Quantity
--------------------------------------------------
Apple               2.30           54
```

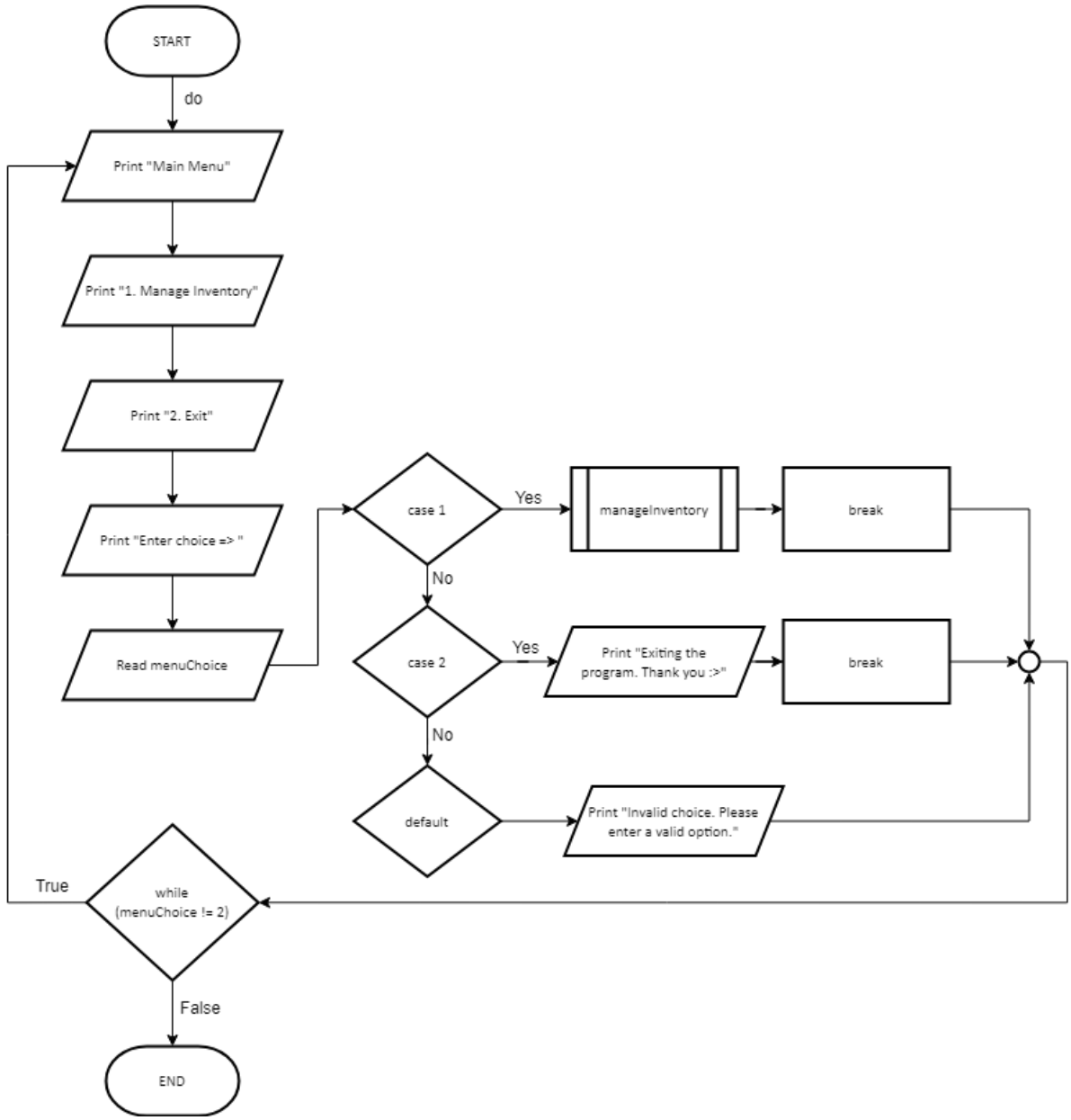
If the user inputs option ‘3’ for the ‘Inventory Management’, the inventory of the grocery store will be displayed in a tabular fashion. The table header consists of “Item Name”, “Price(RM), and “Quantity”. The details of the items in the inventory will be displayed accordingly to the header.

```
1. Add new item
2. Update item quantity
3. Display inventory
4. Exit
Enter choice => 4
Exiting inventory management.

Main Menu:
1. Manage Inventory
2. Exit
Enter choice => 2
Exiting the program. Thank you :>
```

When ‘4’ is inputted in the ‘Inventory Management’, a message will be displayed, “Exiting inventory management.”, and the user will be brought back to the main menu. The user can only fully exit the program by inputting ‘2’, “Exit”, in the main menu where another message will be displayed, “Exiting program. Thank you :>”.

Flowchart



The flowchart showcases the workflow of the program. It starts with a do-while loop, where the program does(do) displays 4 outputs and takes in 1 input for the main menu. The input from the main menu will be read by the 'menuChoice' variable. After an input is entered, a switch statement takes place. If case is '1', the program will proceed with the 'manageInventory' function, where after any final output from the function will bring the program to return to main menu (break). If case is '2', an exit message will be displayed and the program breaks, returning to main menu. If neither '1' or '2' is inputted, it proceeds to the default case where an alert will be displayed for the user to enter a valid option. Finally, proceed to the while-statement, where if the option is '2', ends the program and any option aside from that will loop it back to main menu.

Program Codes

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

#define MAX_ITEMS 50

struct InventoryItem {
    char name[50];
    float price;
    int quantity;
};

```

Two header inclusions are used in this program code. First, '#include stdio.h' on line 1. This line includes standard input-output library, which allows this program to use the 'printf' and 'scanf' functions. Second, '#include string.h' on line 2. This allows for string manipulation functions, for example, string comparison ('strcmp') used in this program.

'#define MAX_ITEMS 50' is a macro definition that creates a macro 'MAX_ITEMS' with a set value of 50, the maximum number of items that can be stored in the inventory.

'struct InventoryItem' is a structure definition that provides a structure named 'InventoryItem' to represent an item in the inventory. It comprises of the essential details of an item in the inventory. This includes the item name, stored in a character array of maximum length of 50 characters ('char name[50];'), the price of the item defined by float to allow floating numbers or decimals ('float price;'), and the quantity of item in stock defined by int to accept whole number values only ('int quantity;').

```c
void manageInventory(struct InventoryItem inventory[], int *numItems) {
    int choice;

    printf("\n1. Add new item\n");
    printf("2. Update item quantity\n");
    printf("3. Display inventory\n");
    printf("4. Exit\n");
    printf("Enter choice => ");
    scanf("%d", &choice);

```

The program has one function named 'manageInventory' that takes two arguments. Firstly, 'inventory[]', an array representing the inventory of the structure 'InventoryItem'. Secondly, '*numItems', a pointer to an integer representing the number of items in the inventory.

The 'manageInventory()' function carries out different inventory management tasks according to the user's input. This includes, adding new items, updating item quantities, displaying inventory, and exiting to the main menu page.

```c
    switch (choice) {
        case 1:
            // Add new item to the inventory
            if (*numItems < MAX_ITEMS) {
                printf("Enter the name of the new item => ");
                scanf("%s", inventory[*numItems].name);
                printf("Enter the price of %s => RM", inventory[*numItems].name);
                scanf("%f", &inventory[*numItems].price);
                printf("Enter the quantity of %s => ", inventory[*numItems].name);
                scanf("%d", &inventory[*numItems].quantity);
                (*numItems)++;
                printf("Item added to the inventory.\n");
            } else {
                printf("Inventory is full. Cannot add more items.\n");
            }
            break;
```

After the user has entered an input for ‘manageInventory()’, the switch statement checks the value of ‘choice’ to execute the specific functionality.

Case 1, when 'choice' is 1, the code is called to add a new item to the inventory. It begins by determining if the number of items ('*numItems') is fewer than the maximum capacity ('MAX_ITEMS'). If there is space, the user is prompted to enter the new item's name, price, and quantity. The corresponding values in the 'inventory[]' array at the index '*numItems' are then updated, and '*numItems' is incremented by one to represent the addition of the new item. Finally, after the item is successfully added to the inventory, a success message is displayed.

If the inventory is full, a notification will appear stating that no more items can be added.

```c
        case 2:
            // Update item quantity
            if (*numItems > 0) {
                char itemName[50];
                int updatedQuantity;

                printf("Enter the name of the item => ");
                scanf("%s", itemName);

                for (int i = 0; i < *numItems; i++) {
                    if (strcmp(inventory[i].name, itemName) == 0) {
                        printf("Enter the new quantity for %s => ", inventory[i].name);
                        scanf("%d", &updatedQuantity);
                        inventory[i].quantity = updatedQuantity;
                        printf("Quantity updated for %s.\n", inventory[i].name);
                        break;
                    }

                    if (i == *numItems - 1) {
                        printf("Item not found in the inventory.\n");
                    }
                }
            } else {
                printf("Inventory is empty. Cannot update quantity.\n");
            }
            break;
```

Case 2, when ‘choice’ is 2, executes code to update the quantity of the existing item. The item names will be saved in the character array 'itemName' for updating. The user is prompted to input the item's name they intend to update. The program iterates over the 'inventory[]' array to locate the provided item name by comparing the input 'itemName' with the names in the inventory using 'strcmp()'. If a match is found, the user is prompted to enter the item's updated quantity, which updates the 'quantity' field of the matched item in the 'inventory[]'. Ends with a quantity update confirmation message.

If the loop completes without locating the item, a message stating that the item was not located in the inventory is displayed.

```
        case 3:
            // Display inventory
            if (*numItems > 0) {
                printf("\nCurrent Inventory:\n");
                printf("-------------------------------------------------\n");
                printf("%-20s%-15s%-10s\n", "Item Name", "Price(RM)", "Quantity");
                printf("-------------------------------------------------\n");

                for (int i = 0; i < *numItems; i++) {
                    printf("%-20s%-15.2f%-10d\n", inventory[i].name, inventory[i].price, inventory[i].quantity);
                }
            } else {
                printf("Inventory is empty.\n");
            }
            break;
```

Case 3, when ‘choice’ is 3, invokes code to display the current inventory. The program prints the title “Current Inventory:” followed by a header for displaying the current inventory details, which include name, price, and quantity. The ‘inventory[]’ array is then iterated through to print the details of each item. The name, price, and quantity of each item is formatted and displayed in a structured manner.

The format specifiers are used for displaying the header to its item value accordingly. ‘%-20s’ indicates a left-aligned string with a width of 20 characters, ‘%-15s’ a width of 15 characters, and ‘%-10s’ a width of 10 characters accordingly.

‘printf’ is used within the loop to format and print the information of each item into a table format. ‘inventory[i].name’, ‘inventory[i].price’, and ‘inventory[i].quantity’ allow for specific information about each item in the inventory[] array to be obtained.

```
        case 4:
            // Exit function
            printf("Exiting inventory management.\n");
            break;

        default:
            printf("Invalid choice. Please enter a valid option.\n");
    }
}
```

Case 4, when ‘choice’ is 4, exits the function, returning to the main menu. A message indicating the exit from the inventory management functionality is displayed.

The default case is executed when the user’s input does not match any of the defined cases in the switch statement. It serves as a way to handle invalid inputs entered by the user and prompts the user to enter a valid option by displaying a message to the user that their input is invalid.

```
int main() {
    struct InventoryItem inventory[MAX_ITEMS];
    int numItems = 0;
    int menuChoice;

    do {
        // Display main menu
        printf("\nMain Menu:\n");
        printf("1. Manage Inventory\n");
        printf("2. Exit\n");
        printf("Enter choice => ");
        scanf("%d", &menuChoice);

        switch (menuChoice) {
            case 1:
                // Manage inventory
                manageInventory(inventory, &numItems);
                break;

            case 2:
                // Exit program
                printf("Exiting the program. Thank you :>\n");
                break;

            default:
                printf("Invalid choice. Please enter a valid option.\n");
        }

    } while (menuChoice != 2);

    return 0;
}
```

The main function sets critical variables required for the program to run. It includes the inventory[] array, designed to store InventoryItem structures to hold inventory data. The 'numItems' variable keeps track of the current quantity of items within the inventory, while 'menuChoice' captures the user's menu selection.

A do-while loop is used in the menu display and user input area to constantly present the main menu to users. This menu prompts users to select between managing the inventory or exiting the program by inputting a numerical choice.

In terms of menu options, selecting '1' triggers a call to the 'manageInventory()' function, which handles various inventory management tasks. Choosing '2' prompts the program to display an exit message and terminates the program. The default case guides users encountering invalid inputs, requesting them to input a valid choice.

The loop's termination condition is determined by the user's option, with the menu being visible until the user chooses to stop the program by selecting '2'. When the program ends, the return statement indicates the successful execution of the main() function and the end of the program's execution.