

Data Structures (DSTR)



DressMe Clothing Marketplace

Student Name:	Wahidul Alam Riyad
TP Number:	TP043338
Intake Code:	UC2F1811IS
Module Code:	CT077-3-2-DSTR
Module Title:	Data Structures
Assessment Title:	DressMe Clothing Marketplace
Assessment Type:	Individual Work
Submission date:	13 th August 2019
Lecturer	Chong Mien May

Table of Contents

1.0	Introduction.....	4
2.0	Flowchart of the DCM System	5
3.0	Snippets of User Interface	6
3.1	Start Up	6
3.2	DressMe Menu	6
3.3	Shop Menu	7
3.4	Display Cloth List.....	7
3.5	Search Cloth by Cloth ID.....	8
3.6	Search Cloth by Cloth Type.....	9
3.7	Add to Cart.....	9
3.8	View Cart	10
3.9	Checkout	11
3.10	Customer Menu	11
3.11	View Order ID	12
3.12	Update Delivery.....	13
3.13	Add Cloth to Shop	13
4.0	Snippets of Data Structures & Algorithms	14
4.1	Display Cart Function	14
4.2	Checkout Function.....	15
4.3	Add a Node to the Orders Linked Lists	16
4.4	Display Order in Detail	16
4.5	Add New Cloth.....	17
4.6	To Read Input Clothes from a File	18
4.7	Input Data into The Node and Then Add It to The Linked List.....	18
4.8	To Sort the Clothes Linked List According to Increasing Price	19
4.9	To Sort the Orders Linked List According Delivery Status and Then Increasing Price	20
4.10	To Search.....	21
4.11	Loading	22

4.12 Switch Case	23
4.13 Update Buyer Details.....	24
5.0 Conclusion	24
References	25

1.0 Introduction

DressMe Clothing Marketplace (DCM) System has been created for DressMe Clothing Sdn. Bhd. Its main purpose is to place an order and view order. Doubly linked lists have been implemented using C++ programming language to build this system. The following functionalities have been implemented:

1. Place an order

- 1.1 View a list of clothes for sale (clothes must be sorted by cost i.e. cheapest first)
- 1.2 Add a new cloth
- 1.3 Search for a type of clothing
- 1.4 Move back and forth between the clothes that are for sale
- 1.5 Select clothes to order
- 1.6 Checkout order

2. View orders

- 2.1 View all orders
- 2.2 Move back and forth between orders (prioritize orders that require delivery)
- 2.3 Modify an order
- 2.4 Delete an order

2.0 Flowchart of the DCM System

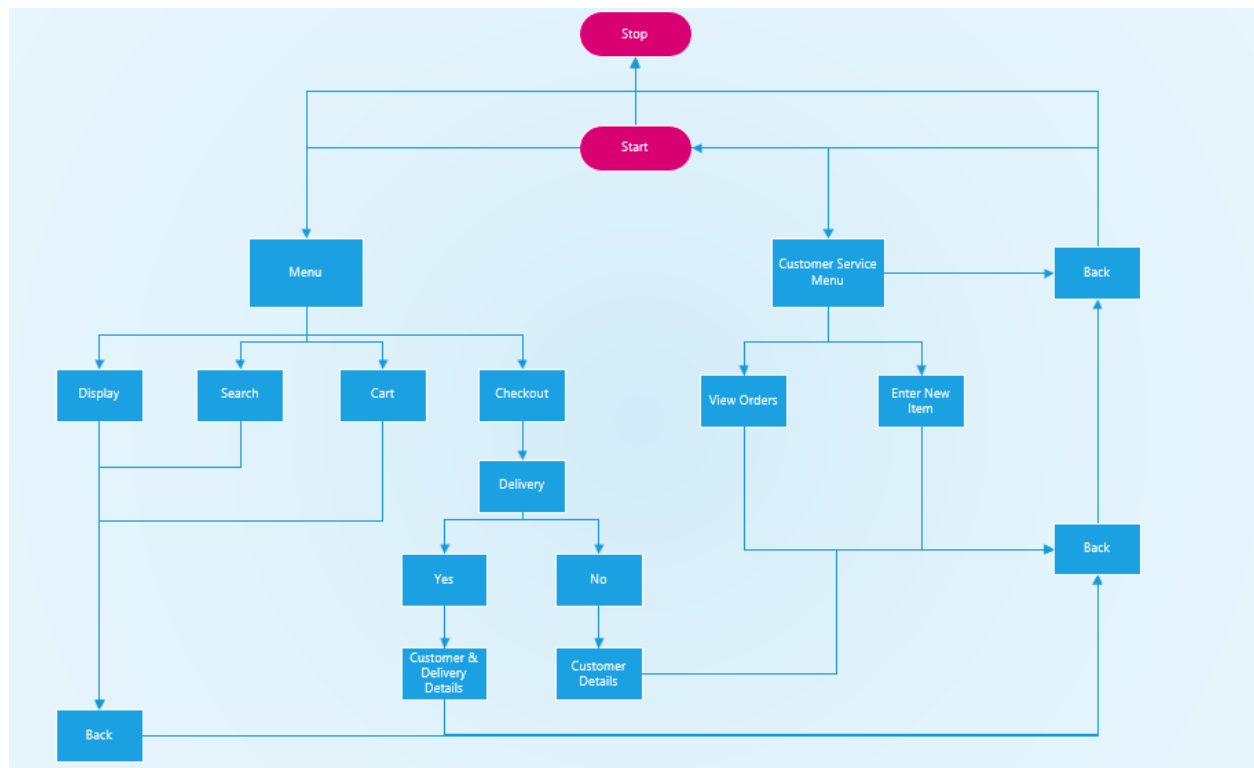


Figure 1 shows the Flowchart of DCM System

Figure 1 shows the Flowchart of DressMe Clothing Marketplace System. It shows the navigation of the system as well.

3.0 Snippets of User Interface

3.1 Start Up

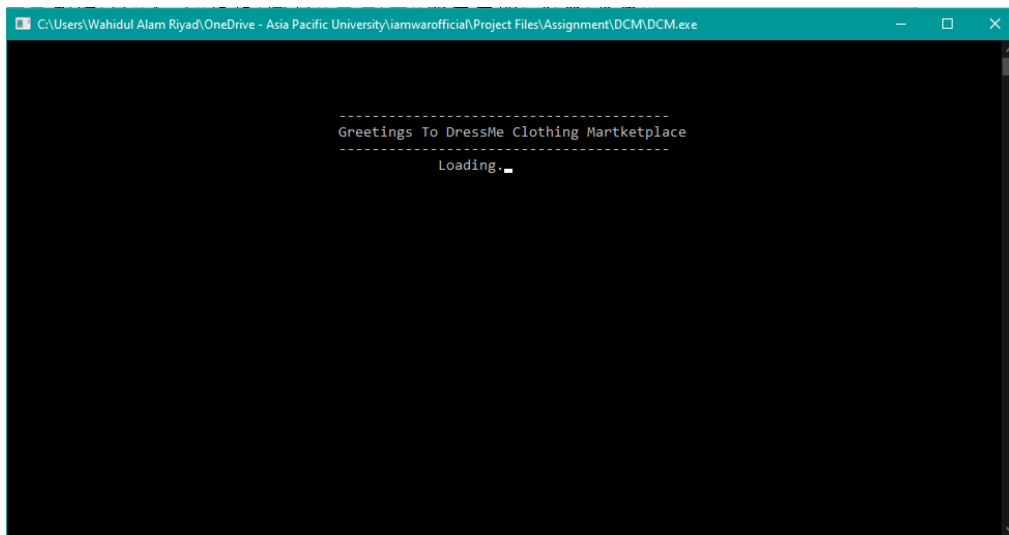


Figure 2 shows Start Up Screen

Figure 2 shows the Start Up Screen of the DCM. It comes with nice animation loading window.

3.2 DressMe Menu

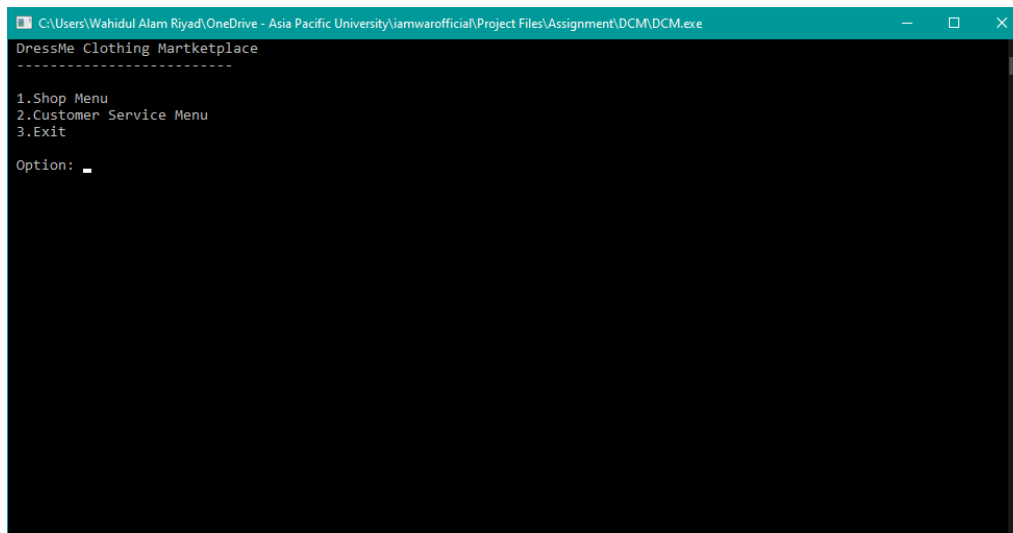


Figure 3 shows DCM Menu

Figure 3 shows the DCM main menu. Here the user has three options. The user can boot into the Shop Menu or the Customer Service Menu.

3.3 Shop Menu

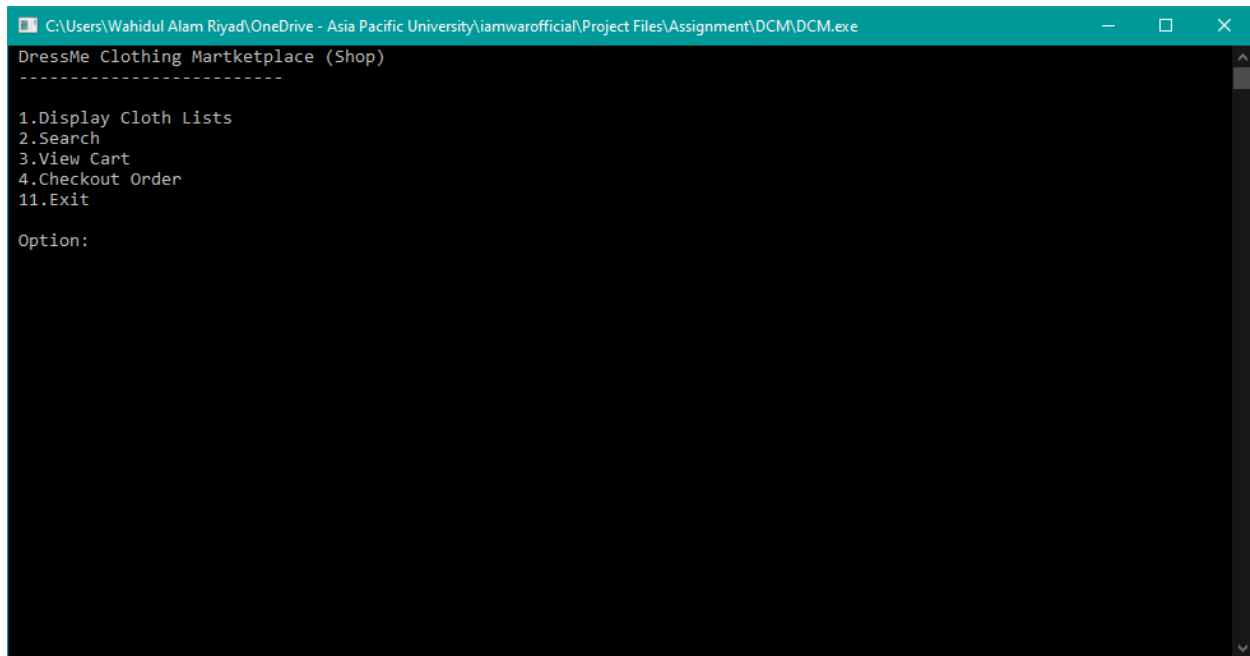


Figure 4 shows the Shop Menu of DCM

Figure 4 shows the Shop Menu. Here the user can view lists of clothes available, search clothes, view cart and checkout order for DCM.

3.4 Display Cloth List

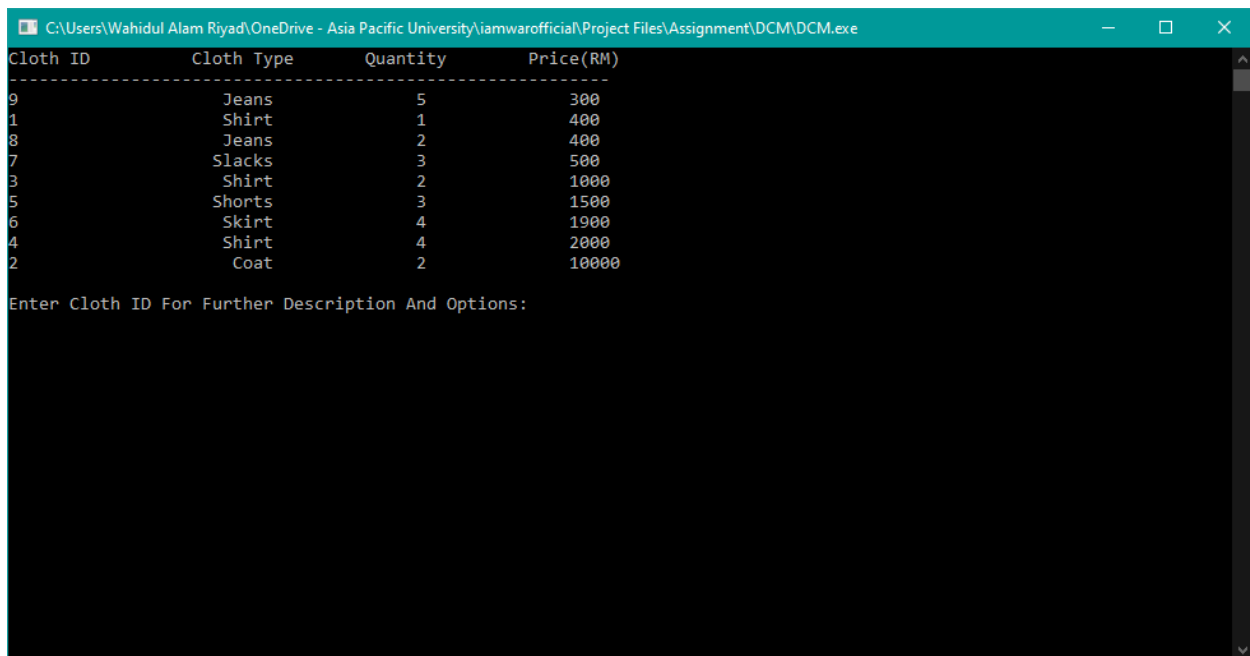


Figure 5 Displays the List of Clothes

Figure 5 shows the lists of clothes available in DCM. Here the clothes are sorted by cost, i.e. cheapest.

3.5 Search Cloth by Cloth ID

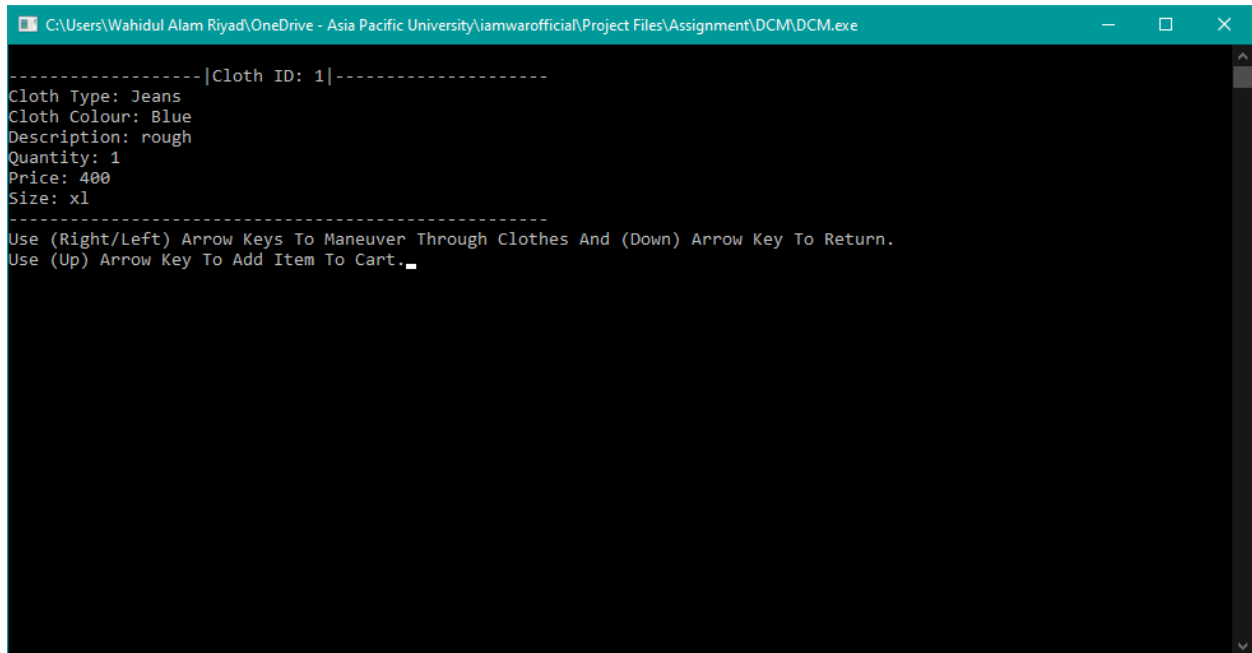


Figure 6 shows Search Cloth by Cloth ID

Figure 6 shows Search Cloth by Cloth ID. Here the user just enter the Cloth ID and it gives all the necessary information regarding that specific cloth.

3.6 Search Cloth by Cloth Type

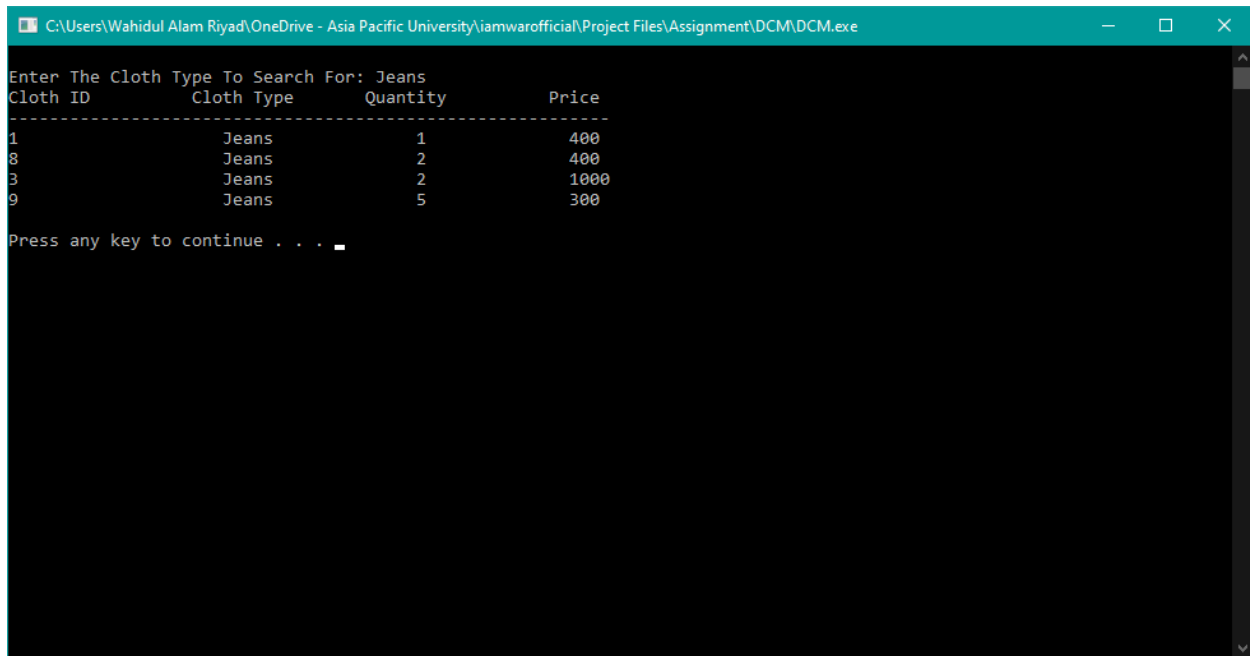


Figure 7 shows Search Cloth by Cloth Type

Figure 7 shows Search Cloth by Cloth Type. Here the user just enters the type of clothes he/she is looking for, and the system will find all the descriptions of the same type of cloth.

3.7 Add to Cart

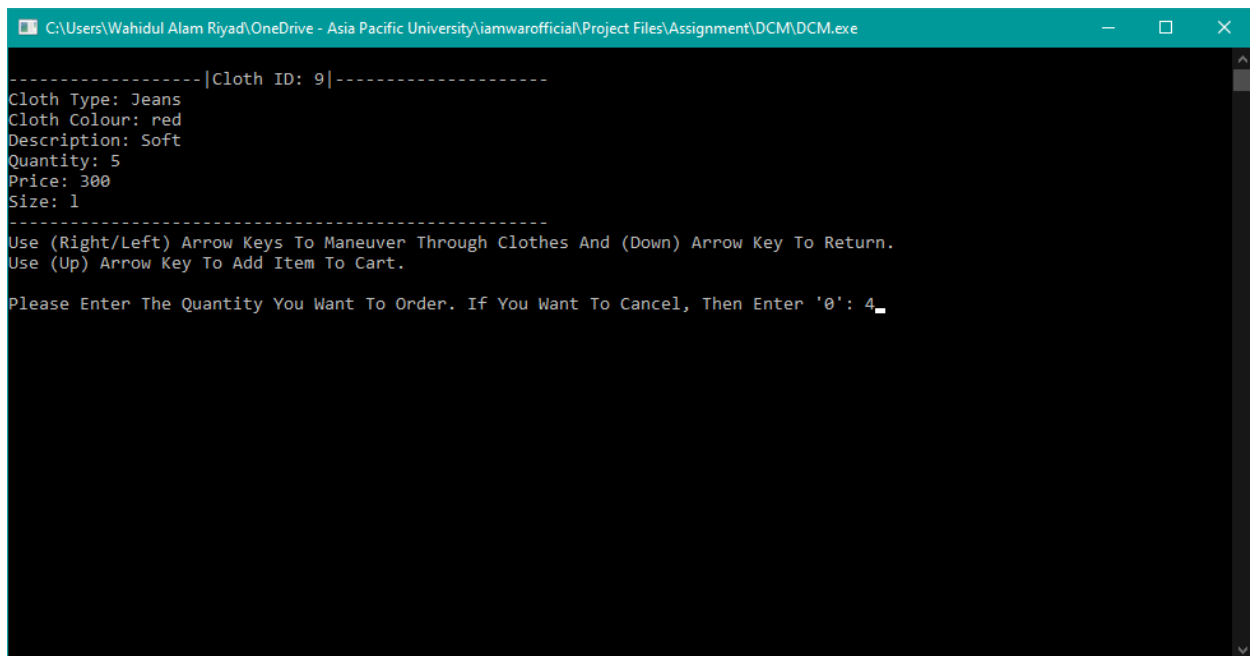
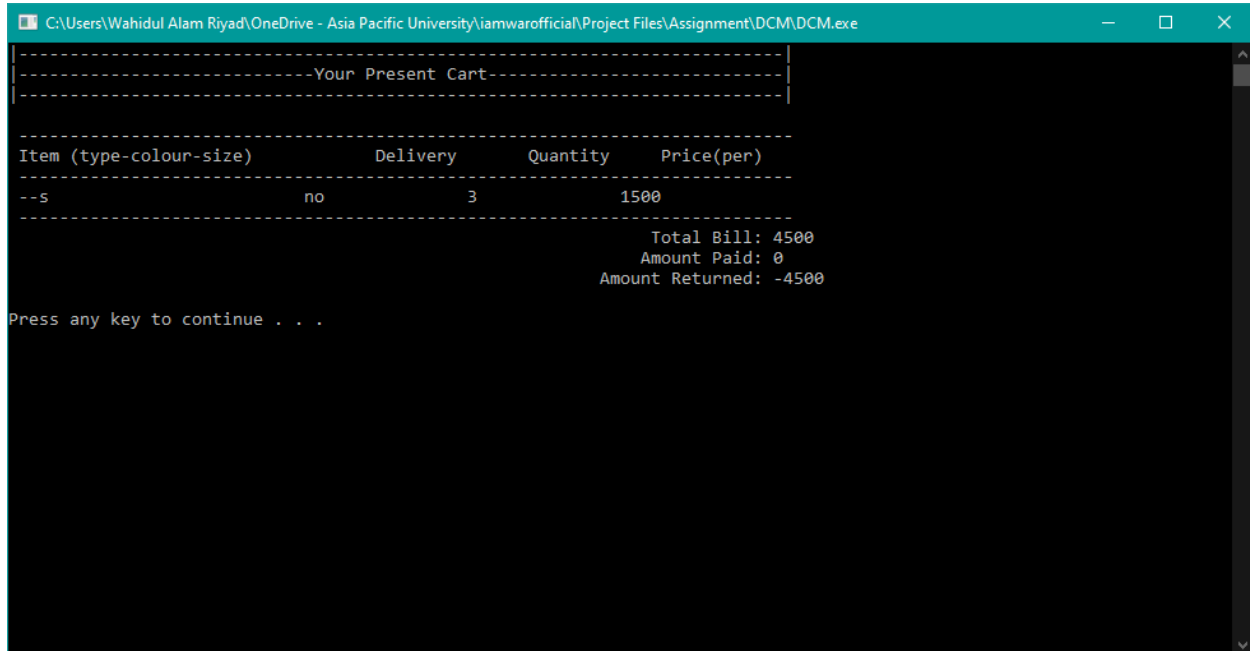


Figure 8 shows the Add to Cart

Figure 8 shows the Add to Cart Function. Here the user can add clothes to their cart by just manoeuvring through clothes. The user has to input the number of quantity he/she is willing to purchase.

3.8 View Cart



```

C:\Users\Wahidul Alam Riyad\OneDrive - Asia Pacific University\iamwarofficial\Project Files\Assignment\DCM\DCM.exe
-----Your Present Cart-----
|
|
|-----|
| Item (type-colour-size) | Delivery | Quantity | Price(per) |
|-----|
| --s | no | 3 | 1500 |
|-----|
|                                     Total Bill: 4500 |
|                                     Amount Paid: 0 |
|                                     Amount Returned: -4500 |
|
| Press any key to continue . . .

```

Figure 9 shows the View Cart

Figure 9 shows the View Cart function that can display all the items that are already added to the cart.

3.9 Checkout

```

C:\Users\Wahidul Alam Riyad\OneDrive - Asia Pacific University\iamwarofficial\Project Files\Assignment\DCM\DCM.exe
-----Your Present Cart-----
-----
Item (type-colour-size)      Delivery      Quantity      Price(per)
-----
--s                          no            3             1500
-----
                                Total Bill: 4500
                                Amount Paid: 0
                                Amount Returned: -4500

Do You Require Delivery (50RM Additionl Cost)? (y/n)

Enter Buyer Name (Must End With A '.'): Wahid.
Enter Buyer Address (Must End With A '.'): Endah.
Enter Buyer Contact Number: 0166901563
Enter Bill Payment (Total Bill is 4550):5000

```

Figure 10 shows the Checkout Window

Figure 10 shows the Checkout Window where the user has to enter delivery details as well. The system asks before whether they do require deliver or not.

3.10 Customer Menu

```

C:\Users\Wahidul Alam Riyad\OneDrive - Asia Pacific University\iamwarofficial\Project Files\Assignment\DCM\DCM.exe
DressMe Clothing Martketplace (Customer Service)
-----
1.View Orders
2.Enter cloth To Shop
11.Exit
Option:

```

Figure 11 shows the Customer Menu

Figure 11 shows the Customer Menu where the user can add new clothes to shop or view order.

3.11 View Order ID

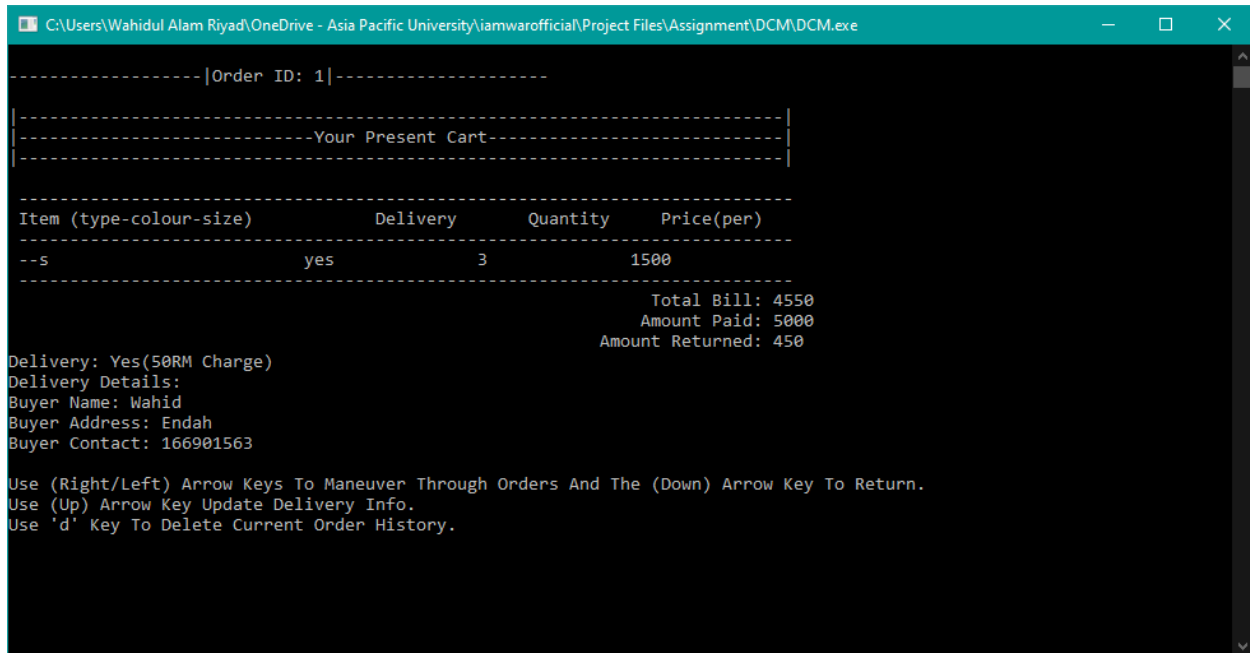


Figure 12 shows Window for Displaying Order ID

Figure 12 shows the details of the orders that has been made. Here the user can also update the delivery information as well.

Figure 14 shows how to add new types of cloth to the system itself. The user has to enter all the necessary information to register new cloth.

4.0 Snippets of Data Structures & Algorithms

4.1 Display Cart Function

```
void disp_cart() //function to display our list in a tabular form
{
    cout << " |-----|\n";
    cout << " |-----Your Present Cart-----|\n";
    cout << " |-----|\n";

    Item *temp_ptr = Item_HeadNode;
    cout << "\n -----";
    cout << "\n Item (type-colour-size)";
    cout << setw(20) << "Delivery";
    cout << setw(15) << "Quantity";
    cout << setw(15) << "Price(per)";
    cout << "\n -----";

    while (temp_ptr != NULL)
    {
        cout << endl << " " << temp_ptr->item_type << "-" << temp_ptr->item_colour << "-"<<temp_ptr->item_size;
        cout << " ";
        if (delivery == 'y')cout << "yes";
        else cout << "no";
        cout << setw(15) << temp_ptr->item_quantity << " ";
        cout << temp_ptr->item_price;
        temp_ptr = temp_ptr->next;
    }

    cout << "\n -----|\n";
    cout << " Total Bill: " << bill;
    cout << "\n Amount Paid: " << payment;
    cout << "\n Amount Returned: " << payment - bill << "\n";
}
```

Figure 15 shows Display Cart Function

This code shows the snippet of the display cart function. Here setw() is used which means Setting field width Using cout in C++ Programming. setw() is library function in C++. setw() is declared inside #include<iomanip>. setw() will set field width. setw() sets the number of characters to be used as the field width for the next insertion operation (C4Learn, 2019).

4.2 Checkout Function

```

void checkout_input()    //checkout function
{
    order_id = Order_ID;

    do
    {
        cout << "\nDo You Require Delivery (50RM Additionl Cost)? (y/n)";
        delivery = _getch();
    } while (delivery != 'y' && delivery != 'n');

    if (delivery == 'y')
    {
        bill = bill + 50;

        cout << "\n\nEnter Buyer Name (Must End With A '.'): ";
        cin.ignore();
        cin.getline(b1.buyer_name, 30, '.');
        cout << "\n\nEnter Buyer Address (Must End With A '.'): ";
        cin.ignore();
        cin.getline(b1.buyer_address, 100, '.');
        cout << "\n\nEnter Buyer Contact Number: ";
        cin.ignore();
        cin >> b1.buyer_no;
        cout << endl;
    }

    while (true)
    {
        cout << "\n\nEnter Bill Payment (Total Bill is " << bill << "):";
        cin >> payment;

        if (payment < bill) continue;
        else break;
    }
}

```

Figure 16 shows the Code for Checkout Function

Figure 16 shows the snippet for the checkout function. This one also shows the delivery function as well.

4.3 Add a Node to the Orders Linked Lists

```
public:

    Cart *cart1;

    void insert_order(Cart *s1) //function used to add a node to the orders linked list
    {
        Cart *cart = new Cart;
        *cart = *s1;
        Cart *temp_ptr = Cart_HeadNode;
        Order_ID++;

        if (Cart_HeadNode == 0)
        {
            Cart_HeadNode = cart;
            cart->next = NULL;
            cart->previous = NULL;
            return;
        }

        while (temp_ptr->next != NULL)
            temp_ptr = temp_ptr->next;

        temp_ptr->next = cart;
        cart->next = NULL;
        cart->previous = temp_ptr;

        return;
    }
}
```

Figure 17 shows How to Add a Node to the Orders Linked Lists

Figure 17 shows the snippet of adding a node in order linked list. The new node is always added before the head of the given Linked List. And newly added node becomes the new head of the Linked List. We are given pointer to a node, and the new node is inserted after the given node. The new node is always added after the last node of the given Linked List (GeeksforGeeks, 2019).

4.4 Display Order in Detail

```
void disp_order(Cart *s1) //function to display whole order in detail and give further options of deleting and updating
{
    //this function uses recursion to allow user to move back and forth along the orders linked list
    system("cls");
    cout << "\n-----" << "|Order ID: " << s1->order_id << " |-----\n\n";

    s1->disp_cart();

    cout << "\n\nUse (Right/Left) Arrow Keys To Maneuver Through Orders And The (Down) Arrow Key To Return.\n";
    cout << "Use (Up) Arrow Key Update Delivery Info.\n";
    cout << "Use 'd' Key To Delete Current Order History.";

    int action;
    do
    {
        action = _getch();
    } while (action != 75 && action != 77 && action != 72 && action != 80 && action != 100);

    if (action == 75 && s1->previous == NULL) disp_order(s1);
    if (action == 75) disp_order(s1->previous);
    if (action == 77 && s1->next == NULL) disp_order(s1);
    if (action == 77 && s1->next != NULL) disp_order(s1->next);
    if (action == 72) { s1->change_buyer_details(); disp_order(s1); }
    if (action == 100) { remove_order(s1); system("cls"); cout << "\n Order Deleted\n"; Sleep(500); cout << "-----"; Sleep(500); return; }
    if (action == 80) return;
}
```

Figure 18 displays Order in Detail

Figure 18 displays the order in detail. If statements are used to see whether where the user action falls. System “cls” is also used to clear the previous output from the command line.

4.5 Add New Cloth

```
void cloth_input(Cloth *c1) //function to input a new cloth
{
    string flag;

    cout << "*****\n";
    cout << "Enter Cloth's Type (Must End With A '.' ) : ";
    cin.ignore();
    cin.getline(c1->cloth_type, 29, '.');

    cout << "Enter Cloth's Colour (Must End With A '.' ) : ";
    cin.ignore();
    cin.getline(c1->cloth_colour, 29, '.');

    cout << "Enter Cloth's Description (Must End With A '.' ) : ";
    cin.ignore();
    cin.getline(c1->cloth_description, 99, '.');

    cout << "Enter Cloth's Quantity: ";
    cin >> c1->cloth_quantity;

    if (c1->cloth_quantity > 0) c1->cloth_availability = 1;
    else c1->cloth_availability = 0;

    do
    {
        cout << "Enter Cloth's Size ('s', 'm', 'l', 'xl' ) : ";
        cin >> c1->cloth_size;

    } while (c1->cloth_size != "s"&& c1->cloth_size != "m"&& c1->cloth_size != "l"&& c1->cloth_size != "xl");

    cout << "Enter Cloth's Price: ";
    cin >> c1->cloth_price;
}
```

Figure 19 shows Addition of New Cloth to the Shop

Figure 19 shows the ask from user to key in the data to add new cloth to the shop. It saves the quantity every time the user purchases a new cloth as well.

4.6 To Read Input Clothes from a File

```
void finput() //function to read input clothes from a file
{
    ifstream fin;
    Cloth *c2 = new Cloth;
    Cloth *temp_ptr;
    Cloth *max_temp;

    fin.open("clothes.txt");
    while (!fin.eof())
    {
        c2 = new Cloth;

        fin >> c2->cloth_id;
        fin.ignore();
        fin.getline(c2->cloth_type, 30, ' ');
        fin.ignore();
        fin.getline(c2->cloth_colour, 30, ' ');
        fin.ignore();
        fin.getline(c2->cloth_description, 100, ' ');
        fin >> c2->cloth_quantity;
        fin >> c2->cloth_price;
        fin >> c2->cloth_size;

        // after storing the data from the file into our cloth structure we now insert the data into our linked list by calling the insert function
        insert_cloth(c2);
    }

    temp_ptr = HeadNode;
    max_temp = HeadNode;
}
```

Figure 20 shows How to Read from Text File

Figure 20 shows the snippet of reading input from text file. ifstream is used and it is saved in the clothes.txt.

4.7 Input Data into The Node and Then Add It to The Linked List

```
void insert_cloth() //function to input data into the node and then add it to the linked list
{
    Cloth *c1 = new Cloth;
    Cloth *temp_ptr = HeadNode;
    ID++;

    if (HeadNode == 0)
    {
        HeadNode = c1;
        cloth_input(c1);
        c1->next = NULL;
        c1->previous = NULL;
        return;
    }

    while (temp_ptr->next != NULL)
        temp_ptr = temp_ptr->next;

    temp_ptr->next = c1;
    cloth_input(c1);
    c1->next = NULL;
    c1->previous = temp_ptr;

    return;
}
```

Figure 21 shows Input Data into the Node

Figure 21 shows how data has been inputted into the node and then add it to the linked list. The new node is always added before the head of the given Linked List. And newly added node becomes the new head of the Linked List. We are given pointer to a node, and the new node is inserted after the given node. The new node is always added after the last node of the given Linked List (GeeksforGeeks, 2019).

4.8 To Sort the Clothes Linked List According to Increasing Price

```
void sort()    //function to sort the clothes linked list according to increasing price
{
    Cloth *temp_ptr;
    Cloth *temp_ptr2;
    Cloth *temp=new Cloth;
    int length = get_length();
    for (int i = 0; i < length - 1; i++)
    {
        temp_ptr = HeadNode;
        temp_ptr2 = temp_ptr->next;

        for (int j = 0; j < length - 1; j++)
        {
            if (temp_ptr->cloth_price > temp_ptr2->cloth_price)
            {
                if (temp_ptr == HeadNode)    //relinking the linked list after a sort step
                {
                    HeadNode = temp_ptr2;
                    temp_ptr->next = temp_ptr2->next;
                    temp_ptr->previous = temp_ptr2;

                    if (temp_ptr2->next != NULL)
                        temp_ptr2->next->previous = temp_ptr;

                    temp_ptr2->next = temp_ptr;
                    temp_ptr2->previous = NULL;
                }
            }
            else
            {
                temp_ptr->next = temp_ptr2->next;    //relinking the linked list after a sort step

                if (temp_ptr2->next != NULL)
                    temp_ptr2->next->previous = temp_ptr;

                temp_ptr->previous->next = temp_ptr2;
            }
        }
    }
}
```

Figure 22 shows how to sort the clothes according to price

Figure 22 shows the sorting algorithm that is used to sort the clothes according to the price. A doubly linked list consists of a set of sequentially linked records called nodes. Each node contains three fields: one data field and two link fields; references to the previous and to the next node in the sequence of nodes field. In case of sorted doubly linked list, according to the values of the sorted data field values the linked list will remain sorted (TutorialsPoint, 2019).

4.9 To Sort the Orders Linked List According Delivery Status and Then Increasing Price

```
void sort_orders()    //function to sort the orders linked list according delivery status and then increasing price
{
    Cart *temp_ptr;
    Cart *temp_ptr2;
    Cart *temp= new Cart;
    int length = get_order_length();
    for (int i = 0; i < length - 1; i++)
    {
        temp_ptr = Cart_HeadNode;
        temp_ptr2 = temp_ptr->next;

        for (int j = 0; j < length - 1; j++)
        {
            if ((temp_ptr->bill > temp_ptr2->bill) || (temp_ptr->delivery == 'n' && temp_ptr2->delivery == 'y'))
            {
                if (temp_ptr == Cart_HeadNode)    //relinking the linked list after a sort step
                {
                    Cart_HeadNode = temp_ptr2;
                    temp_ptr->next = temp_ptr2->next;
                    temp_ptr->previous = temp_ptr2;

                    if (temp_ptr2->next != NULL)
                        temp_ptr2->next->previous = temp_ptr;

                    temp_ptr2->next = temp_ptr;
                    temp_ptr2->previous = NULL;
                }
                else
                {
                    temp_ptr->next = temp_ptr2->next;    //relinking the linked list after a sort step

                    if (temp_ptr2->next != NULL)
                        temp_ptr2->next->previous = temp_ptr;
                }
            }
        }
    }
}
```

Figure 23 shows the Order Linked List

Figure 23 shows the snippet of sorting the order linked list according to delivery status and increasing price. Sorting algorithms is also used here.

4.10 To Search

```

void search() //function to search
{
    Cloth *temp_ptr = HeadNode;
    char type[30];
    bool found = 0;

    cout << "\nEnter The Cloth Type To Search For: ";
    cin >> type;

    cout << "Cloth ID";
    cout << setw(20) << "Cloth Type";
    cout << setw(15) << "Quantity";
    cout << setw(15) << "Price";
    cout << "\n-----";

    while (temp_ptr != NULL)
    {
        if (!strcmp(temp_ptr->cloth_type, type))
        {
            found = 1;
            cout << endl << temp_ptr->cloth_id << setw(25);
            cout << temp_ptr->cloth_type << setw(30);
            cout << setw(15) << temp_ptr->cloth_quantity << " ";
            cout << temp_ptr->cloth_price;

            temp_ptr = temp_ptr->next;
        }

        if (temp_ptr == NULL && found == 0)
        {
            cout << endl << "Invalid Cloth Type";
        }
    }
}

```

Figure 24 shows the Search Function

Figure 24 shows the Search Function where the user has to key in the type of cloth to find the list of clothes.

4.12 Switch Case

```
system("cls");
cout << " DressMe Clothing Martketplace (Shop)\n";
cout << " -----\n\n";
cout << " 1.Display Cloth Lists\n";
cout << " 2.Search\n";
cout << " 3.View Cart\n";
cout << " 4.Checkout Order\n";
cout << " 11.Exit\n\n";
cout << " Option: ";
cin >> choice2;

switch (choice2)
{
case 1: system("cls");
      sort();
      disp_list();
      break;

case 2: system("cls");
      search(); cout << endl << endl;
      system("PAUSE");
      break;

case 3: system("cls");
      view_cart();
      system("pause");
      break;

case 4: system("cls");
      checkout();
      choice2 = 11;
      break;
}
```

Figure 26 shows the Implementation of using Switch Case

Figure 26 the switch case scenario to manoeuvre different options for the user.

4.13 Update Buyer Details

```
void change_buyer_details() //function to update buyer details
{
    if (delivery == 'n')
    {
        cout << "\n\nYou Cannot Modify The Buyer Details In A Non Delivery Order.\n";
        system("pause");
        return;
    }

    int choice;
    while (true)
    {
        cout << "\n\n 1.Modify Name\n";
        cout << " 2.Modify Address\n";
        cout << " 3.Modify Contact Number\n";
        cout << " 4.Exit";
        cout << " choice: ";
        cin >> choice;

        if (choice == 1)
        {
            cout << "\nEnter Name (Must End With A '.'): ";
            cin.ignore();
            cin.getline(b1.buyer_name, 30, '.');
        }

        if (choice == 2)
        {
            cout << "\nEnter Address (Must End With A '.'): ";
            cin.ignore();
            cin.getline(b1.buyer_address, 100, '.');
        }

        if (choice == 3)
        {
            cout << "\nEnter Phone Number:";
            cin >> b1.buyer_no;
        }
    }
}
```

Figure 27 shows Modify Buyer Details

Figure 27 shows the options to modify buyer details, in terms of name or address or phone number.

5.0 Conclusion

A Doubly Linked List (DLL) contains an extra pointer, typically called previous pointer, together with next pointer and data which are there in singly linked list. A DLL can be traversed in both forward and backward direction. The delete operation in DLL is more efficient if pointer to the node to be deleted is given. We can quickly insert a new node before a given node. In singly linked list, to delete a node, pointer to the previous node is needed. To get this previous node, sometimes the list is traversed. In DLL, we can get the previous node using previous pointer.

References

C4Learn, 2019. *C4Learn*. [Online]

Available at: <http://www.c4learn.com/cplusplus/cpp-setw-setting-field-width/>

GeeksforGeeks, 2019. *GeeksforGeeks*. [Online]

Available at: <https://www.geeksforgeeks.org/linked-list-set-2-inserting-a-node/>

TutorialsPoint, 2019. *TutorialsPoint*. [Online]

Available at: <https://www.tutorialspoint.com/cplusplus-program-to-implement-sorted-doubly-linked-list>