UOW MALAYSIA KDU PENANG UNIVERSITY COLLEGE

PART OF THE UNIVERSITY OF WOLLONGONG AUSTRALIA GLOBAL NETWORK

ASSIGNMENT COVER PAGE

Programme	**Course Code and Title**
Diploma in Information Technology	DOP1254 Fundamentals of Object Oriented Programming
Student's name / student's id	**Lecturer's name**
• 0204677 LIM ZHE YUAN • 0205096 THOR WEN ZHENG • 0205430 TAN PENG HENG	Ms. Tan Phit Huan

Date issued	**Submission Deadline**	**Indicative Weighting**
Week 6 – 03/05/2021	Week 11 – 09/04/2021	20%
Assignment 2 title	Product management system	

This assessment assesses the following course learning outcomes

# as in Course Guide	**UOWM KDU Penang University College Learning Outcome**
CLO2	Apply modularization and array in programming.
CLO4	Apply object oriented programming concepts in software development.

Student's declaration

I certify that the work submitted for this assignment is my own and research sources are fully acknowledged.

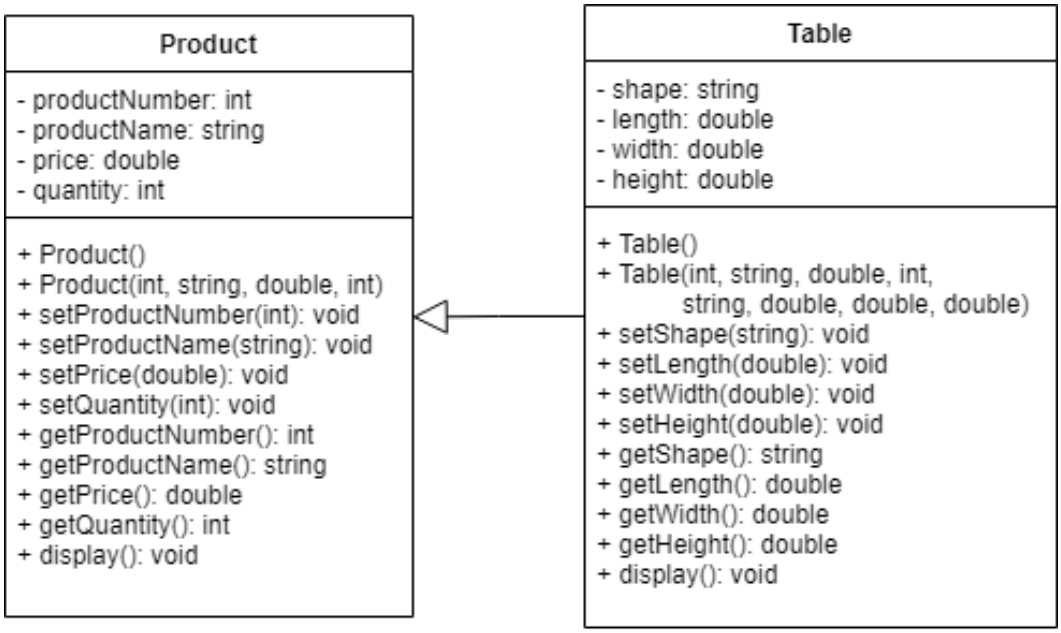
Student's signature: LIM ZHE YUAN
THOR WEN ZHENG
TAN PENG HENG

Submission Date: 9 April 2021

Table of Contents

Main Report

1. Class Diagram



2. Complete Program

- **Product Class**

```
#ifndef PRODUCT_H
#define PRODUCT_H
#include <iostream>
using namespace std;

// Product class (superclass)
class Product {
	private:
		int productNumber, quantity;
		string productName;
		double price;
	public:
		// Constructors for Product
		Product() {
			productName = "Name Unknown";
		}
		Product(int num, string name, double pr, int qty) {
			productNumber = num;
			productName = name;
			price = pr;
			quantity = qty;
		}
		// Getter and setter methods for Product
		void setProductNumber(int num) {
			productNumber = num;
		}
		void setProductName(string n) {
			productName = n;
		}
		void setPrice(double p) {
			price = p;
		}
		void setQuantity(int qty) {
			quantity = qty;
		}
		int getProductNumber() {
			return productNumber;
		}
		string getProductName() {
			return productName;
		}
		double getPrice() {
			return price;
		}
		int getQuantity() {
			return quantity;
```

```
        }
        // Query methods for Product
        void display() {
            cout << "Product Number: " << productNumber << endl
                 << "Product Name: " << productName << endl
                 << "Price: " << price << endl
                 << "Quantity: " << quantity << endl;
        }
};
#endif
```

- **Table Class**

```
#ifndef TABLE_H
#define TABLE_H
#include "Product.h"
#include <iostream>
using namespace std;
// Table class (subclass)
class Table: public Product {
    private:
        string shape;
        double length, width, height;
    public:
        // Constructors for Table
        Table() {
            length = 1.0;
            width = 1.0;
            height = 1.0;
        }
        Table(int num, string name, double pr, int qty, string sh, double len, double w,
double h): Product(num, name, pr, qty) {
            shape = sh;
            length = len;
            width = w;
            height = h;
        }
        // Getter and setter methods for Table
        void setShape(string s) {
            shape = s;
        }
        void setLength(double len) {
            length = len;
        }
        void setWidth(double w) {
            width = w;
        }
        void setHeight(double h) {
            height = h;
```

```
        }
        string getShape() {
                return shape;
        }
        double getLength() {
                return length;
        }
        double getWidth() {
                return width;
        }
        double getHeight() {
                return height;
        }
        // Query methods for Table
        void display() {
                Product::display();
                cout << "Shape: " << shape << endl
                        << "Length: " << length << endl
                        << "Width: " << width << endl
                        << "Height: " << height << endl;
        }
};
#endif
```

- **Driver Program**

```
#include <iostream>
#include <iomanip>    // For fixed, showpoint, setprecision, to show values with decimal points
#include <cctype>     // For toupper
#include <limits>     // For using numeric_limits in cin.ignore
#include "Table.h"     // includes Table header file, which also includes Product header file
using namespace std;

// Function prototypes
int showMenu();
void addRecord(Table tables[]);
void editRecord(Table tables[]);
void searchRecord(Table tables[]);
void checkRecords(Table tables[]);
int checkCinState();

// Global variables
const int SIZE = 100;
int index = 0;

int main() {
        // Initialize Table array
        Table tableArr[SIZE];
        for (int i = 0; i < SIZE; i++) {
```

```
		tableArr[i].setProductNumber(0);
	}

	// Variables
	int menuInput;
	char exitInput;
	bool isEnded = false;

	cout << "Welcome to Product Management System\n\n";
	cout << "* * * PLEASE MAXIMIZE CONSOLE WINDOW FOR BEST EXPERIENCE * *
*\n\n";
	do {
		// Show menu and get menu input
		menuInput = showMenu();
		// Check menu input and redirect to selected function
		if (menuInput == 1) {
			addRecord(tableArr);
		} else if (menuInput == 2) {
			editRecord(tableArr);
		} else if (menuInput == 3) {
			searchRecord(tableArr);
		} else if (menuInput == 4) {
			checkRecords(tableArr);
		} else if (menuInput == 5) {
			// Exit
			do {
				cout << "Are you sure? (Y/N): ";
				cin >> exitInput;
				exitInput = toupper(exitInput);
				cin.ignore(numeric_limits<streamsize>::max(), '\n');

				if (exitInput == 'Y')
					isEnded = true;
			} while (exitInput != 'Y' && exitInput != 'N');
		}
	} while (!isEnded);

	return 0;
}

int showMenu() {
	// Variables
	int input;
	bool menuBool = true;

	do {
		// Print menu
		cout << endl;
		cout.fill('=');
		cout << setw(50) << right << "=\n";
		cout.fill(' ');
```

```
		cout << setw(27) << right << "MENU\n";
		cout.fill('=');
		cout << setw(50) << right << "=\n";
		cout.fill(' ');
		cout << "\n		" << "1. Add record\n"
			<< "		" << "2. Edit record\n"
			<< "		" << "3. Search record\n"
			<< "		" << "4. Check records\n"
			<< "		" << "5. Exit\n\n";
		// Get selection input
		cout << "Selection: ";
		cin >> input;
		cin.ignore(numeric_limits<streamsize>::max(), '\n');
		// Check cin state
		if (cin.fail()) {
			cin.clear();
			cin.ignore(numeric_limits<streamsize>::max(), '\n');
		}
		// Check selection input
		if (input < 1 || input > 5) {
			cout << "\n* * * Invalid input. Valid inputs: 1, 2, 3, 4, 5 * * * \n";
			menuBool = true;
		} else {
			menuBool = false;
		}
	} while (menuBool);

	return input;
}

void addRecord(Table tables[]) {
	// Variables
	int num, quantity, error;
	string name, shape;
	double price, length, width, height;

	// Check if Table array is full
	if (index == SIZE) {
		cout << "\n\n* * * MAXIMUM AMOUNT OF RECORDS REACHED, UNABLE TO
ADD NEW RECORD * * *\n";
		cout << "\n* * * RETURNING TO MENU * * *\n";
	}
	else {
		// Get input from user
		cout << "\n\nPlease key in details of the Table.\n";
		cout.fill('_');
		cout << setw(70) << "_\n";
		// Get Product Number
		do {
			cout << "Product Number: ";
			cin >> num;
```

```
                        error = checkCinState();
                        // Check if product already exists in array
                        for (int i = 0; i < SIZE; i++) {
                                if (num == tables[i].getProductNumber() &&
tables[i].getProductNumber() > 0) {
                                        cout << "* * * Table record already exists in the system. * *
*\n";
                                        error = 69;
                                        break;
                                }
                        }
                        // Check for valid product number
                        if ((num < 1 || num > 9999) && error == 0)
                                cout << "* * * Product numbers range from 1 to 9999 only. * * *\n";
                } while (num < 1 || num > 9999 || error == 69);
                // Get Product Name
                do {
                        cout << "Product Name: ";
                        getline(cin, name);
                        if (name == "" || name == " ")
                                cout << "* * * Name input cannot be empty. * * *\n";
                } while (name == "" || name == " ");
                // Get Price
                do {
                        cout << "Price: ";
                        cin >> price;
                        error = checkCinState();
                        if (price < 0)
                                cout << "* * * Only positive numbers allowed. * * *\n";
                } while (price < 0 || error == 69);
                // Get Quantity
                do {
                        cout << "Quantity: ";
                        cin >> quantity;
                        error = checkCinState();
                        if (quantity < 0)
                                cout << "* * * Only positive numbers allowed. * * *\n";
                } while (quantity < 0 || error == 69);
                // Get Shape
                do {
                        cout << "Shape: ";
                        getline(cin, shape);
                        if (shape == "" || shape == " ")
                                cout << "* * * Shape input cannot be empty. * * *\n";
                } while (shape == "" || shape == " ");
                // Get Length
                do {
                        cout << "Length: ";
                        cin >> length;
                        error = checkCinState();
                        if (length < 1 && error == 0)
```

```
                        cout << "* * * Length must be greater than 0. * * *\n";
                } while (length < 1 || error == 69);
                // Get Width
                do {
                        cout << "Width: ";
                        cin >> width;
                        error = checkCinState();
                        if (width < 1 && error == 0)
                                cout << "* * * Width must be greater than 0. * * *\n";
                } while (width < 1 || error == 69);
                // Get Height
                do {
                        cout << "Height: ";
                        cin >> height;
                        error = checkCinState();
                        if (height < 1 && error == 0)
                                cout << "* * * Height must be greater than 0. * * *\n";
                } while (height < 1 || error == 69);

                // Confirm if user really wants to add new Table
                char conf;
                cout << endl;
                do {
                        cout << "Confirm new Table? (Y/N): ";
                        cin >> conf;
                        conf = toupper(conf);
                        cin.ignore(numeric_limits<streamsize>::max(), '\n');
                } while (conf != 'Y' && conf != 'N');

                // If Yes, add Table record; if No, do nothing and return to menu
                if (conf == 'Y') {
                        cout << "\n+ + + New Table record has been added. + + +\n";

                        /* Create new Table object with assigned values using constructor,
                           then assign Table object to corresponding index in tables array */
                        Table tableObj(num, name, price, quantity, shape, length, width, height);
                        tables[index] = tableObj;

                        // Increase index
                        index++;

                        // Notify user if Table array is full
                        if (index == SIZE)
                                cout << "\n* * * MAXIMUM AMOUNT OF RECORDS REACHED *
* *\n\n";
                } else if (conf == 'N') {
                        cout << "\n* * * Process cancelled. Returning to menu. * * *\n\n";
                }
        }
}
```

```
void editRecord(Table tables[]) {
	// Variables
	int num, error, matchedIndex;
	double length, width, height;

	// Prompt user for product number
	cout << "\n\nPlease key in the product number.\n";
	cout.fill('_');
	cout << setw(70) << right << "_\n";
	do {
		cout << "Product Number: ";
		cin >> num;
		error = checkCinState();
		if ((num < 1 || num > 9999) && error == 0)
			cout << "* * * Invalid product number. Product number ranges from 1 -
9999. * * *\n";
	} while (num < 1 || num > 9999 || error == 69);

	// Check if product exists in records
	for (int i = 0; i < SIZE; i++) {
		if (num == tables[i].getProductNumber()) {
			matchedIndex = i;
			break;
		}
		if (i == 99) {
			cout << "\n* * * No matching product found. Returning to menu.* * *\n\n";
			return;
		}
	}

	// Show current values of selected product
	cout << "\nCurrent values of PRODUCT " << tables[matchedIndex].getProductNumber()
<< ":\n";
	cout << setw(70) << right << "_\n";
	tables[matchedIndex].display();

	// Prompt user to key in new values
	cout << "\nPlease key in new length, width and height values for PRODUCT " <<
tables[matchedIndex].getProductNumber() << ".\n";
	cout << setw(70) << right << "_\n";
	do {
		cout << "Length: ";
		cin >> length;
		error = checkCinState();
		if (length < 1 && error == 0)
			cout << "* * * Length must be greater than 0. * * *\n";
	} while (length < 1 || error == 69);
	do {
		cout << "Width: ";
		cin >> width;
		error = checkCinState();
```

```
			if (width < 1 && error == 0)
				cout << "* * * Length must be greater than 0. * * *\n";
		} while (width < 1 || error == 69);
		do {
			cout << "Height: ";
			cin >> height;
			error = checkCinState();
			if (height < 1 && error == 0)
				cout << "* * * Length must be greater than 0. * * *\n";
		} while (height < 1 || error == 69);

		// Confirm if user really wants to change value
		char conf;
		cout << endl;
		do {
			cout << "Confirm new values? (Y/N): ";
			cin >> conf;
			conf = toupper(conf);
			cin.ignore(numeric_limits<streamsize>::max(), '\n');
		} while (conf != 'Y' && conf != 'N');

		// If Yes, change values; if No, do nothing and return to menu
		if (conf == 'Y') {
			cout << "\n+ + + New values have been applied. + + +\n\n";
			tables[matchedIndex].setLength(length);
			tables[matchedIndex].setWidth(width);
			tables[matchedIndex].setHeight(height);
		} else if (conf == 'N') {
			cout << "\n* * * Process cancelled. Returning to menu. * * *\n\n";
		}
}

void searchRecord(Table tables[]) {
		// Variables
		int num, error;

		// Get product number from user
		cout << "\n\nPlease key in the product number.\n";
		cout.fill('_');
		cout << setw(70) << "_\n";
		do {
			cout << "Product Number: ";
			cin >> num;
			error = checkCinState();
			if ((num < 1 || num > 9999) && error == 0)
				cout << "* * * Invalid product number. Product number ranges from 1 -
9999. * * *\n";
		} while (num < 1 || num > 9999 || error == 69);

		// Search for product in tables array
		for (int i = 0; i < SIZE; i++) {
```

```
			if (num == tables[i].getProductNumber()) {
				cout << "\nProduct found. Displaying product details: \n";
				cout.fill('_');
				cout << setw(70) << "_\n";
				tables[i].display();
				break;
			}
			if (i == 99)
				cout << "\n* * * No matching product found. Returning to menu. * * *\n\n";
		}
}

void checkRecords(Table tables[]) {
		// Variables
		bool isEmpty = false;

		// Check if tables array is empty
		if (tables[0].getProductNumber() == 0)
			isEmpty = true;

		if (isEmpty) {
			// Indicate empty records
			cout << endl;
			cout.fill('=');
			cout << setw(126) << "=\n";
			cout.fill(' ');
			cout << setw(73) << "NO RECORDS";
			cout.fill('=');
			cout << setw(126) << left << "\n=";
			cout << endl;
		} else {
			// Print table header
			cout << endl;
			cout.fill('=');
			cout << setw(182) << right << "=\n";
			cout.fill(' ');
			cout << setw(6) << left << "No.";
			cout << setw(20) << left << "Product Number";
			cout << setw(36) << left << "Product Name";
			cout << setw(20) << left << "Price";
			cout << setw(20) << left << "Quantity";
			cout << setw(20) << left << "Shape";
			cout << setw(20) << left << "Length";
			cout << setw(20) << left << "Width";
			cout << setw(20) << left << "Height";
			cout << endl;
			cout.fill('=');
			cout << setw(182) << right << "=\n";
			// Print table contents
			cout.fill(' ');
			cout << fixed << showpoint << setprecision(2);
```

```
		for (int i = 0; i < SIZE; i++) {
			if (tables[i].getProductNumber() > 0) {
				cout << i + 1 << ".    ";
				cout << setw(20) << left << tables[i].getProductNumber();
				cout << setw(36) << left << tables[i].getProductName();
				cout << setw(20) << left << tables[i].getPrice();
				cout << setw(20) << left << tables[i].getQuantity();
				cout << setw(20) << left << tables[i].getShape();
				cout << setw(20) << left << tables[i].getLength();
				cout << setw(20) << left << tables[i].getWidth();
				cout << setw(20) << left << tables[i].getHeight();
			} else {
				break;
			}
			cout << endl;
		}
		cout << endl;
	}
}

int checkCinState() {
	// Used for checking cin fail state
	// Returns 69 if cin failed, returns 0 if no error
	if (cin.fail()) {
		cout << "* * * Invalid input * * *" << endl;
		cin.clear();
		cin.ignore(numeric_limits<streamsize>::max(), '\n');
		return 69;
	} else {
		cin.ignore(numeric_limits<streamsize>::max(), '\n');
		return 0;
	}
}
```

3. Description of the Program

- **Introduction & Main Menu**

This system is a product management system for a furniture manufacturing company to handle new and existing records of table products. When the system starts, a welcome message is displayed to the user and then the main menu of the system is also displayed. The system also notifies the user to maximize the console window to ensure that the user interface is not hindered for the best user experience. The main menu shows 5 options, the first option is “Add record”, the second option is “Edit record”, the third and fourth options are “Search record” and “Check records” respectively, and the last option is “Exit”. These 5 options are basically the main functions of the system.

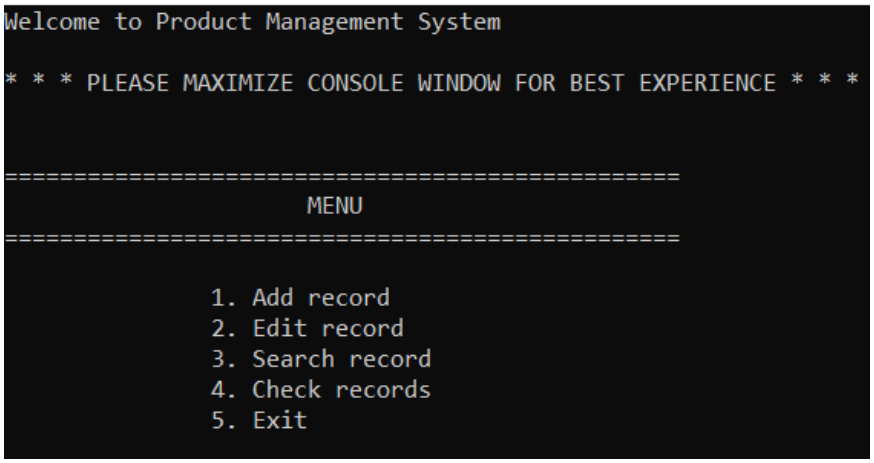


Figure 1.0: Welcome message and main menu of the system.

Immediately after the main menu is shown, the user will be prompted for a “Selection” input, which represents the main function that the user wants to select. For the “Selection” input, the system only accepts integer values from 1 to 5; if the user inputs any other value, the system will notify the user that the input is invalid and remind the user what the acceptable inputs are, then it redisplays the menu and prompts the user for another input. The system will keep prompting the user for an input until the user enters a valid input value. Upon receiving a valid input, the system redirects the user to the specific system function that they selected.

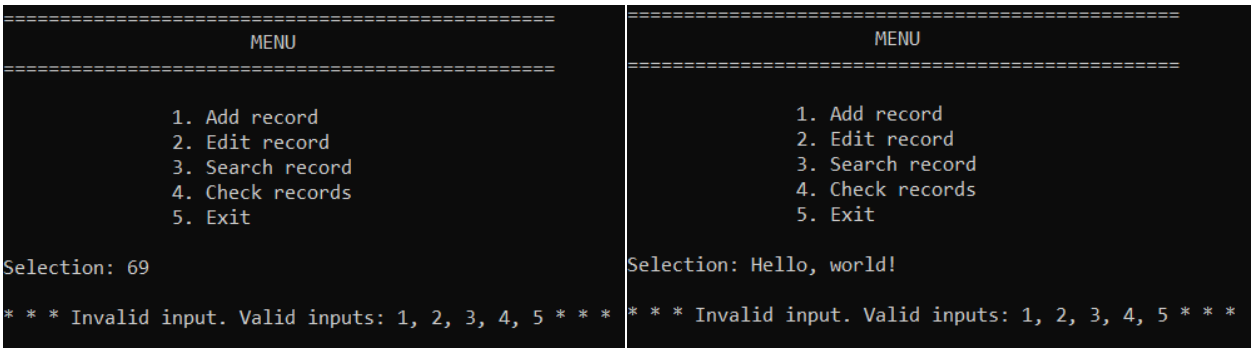


Figure 1.1: Invalid inputs are rejected; system repeatedly prompts user for valid input.

- **Add Record**

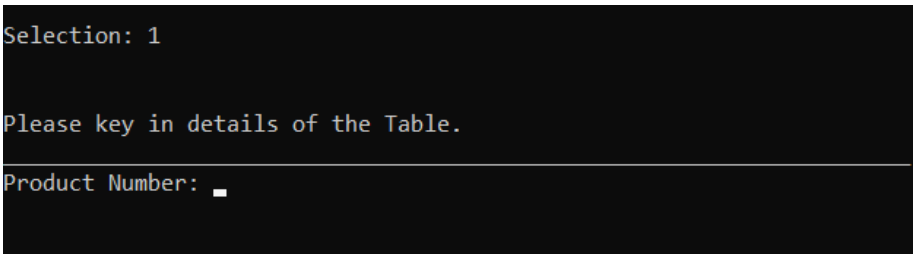


Figure 2.0: User inputs 1, system redirects user to the “Add record” function.

In the case that the user inputs “1” in the menu, the system redirects the user to the “Add record” function. The purpose of the “Add record” function is to allow the user to add new table records into the system. In this function, the system first prompts the user to enter the details of the table product, including the product number, product name, price, quantity, shape, and its dimensions such as width, length, and height. The user must ensure that the product number that they entered is unique, else the system will notify the user that an existing table record already holds the product number that they have entered. The system will notify the user about any invalid inputs that they encounter and allow them to re-enter their inputs again.

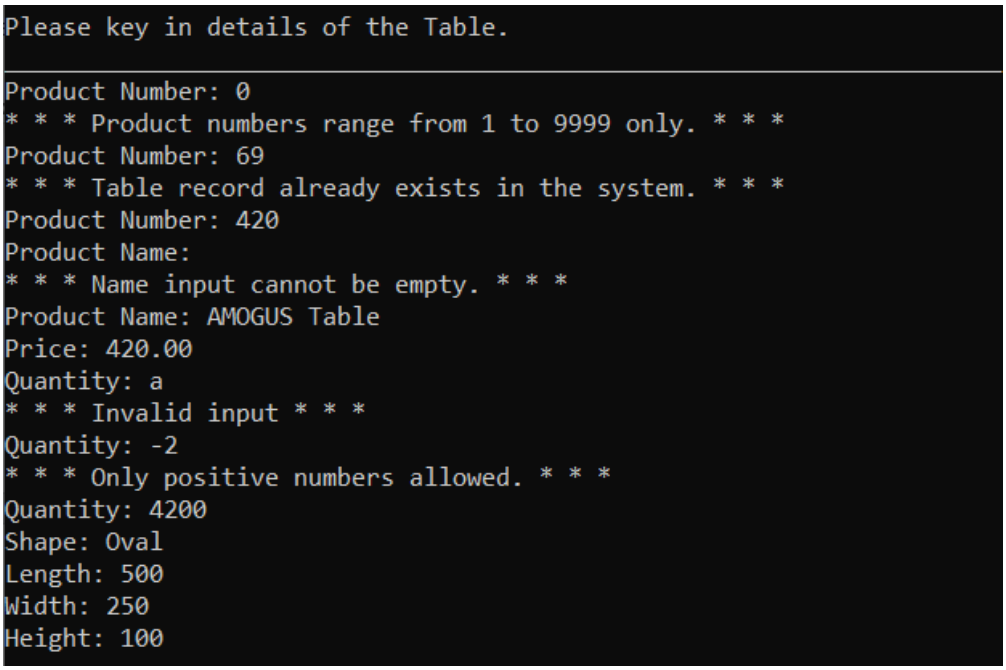


Figure 2.1: Process of entering details of table record

After entering the details of the product, the system asks the user for confirmation on whether the table record should be created or not. If the user enters ‘Y’, the system will print out a message saying that the table record has been successfully added into the system. Conversely, if the user enters ‘N’, the system will print out a message saying that the record creation process is cancelled. Regardless of the choice, the user will be redirected back to the main menu where they will be prompted for a selection input again. If the user enters neither ‘Y’ nor ‘N’, the system prompts the user for another confirmation input.

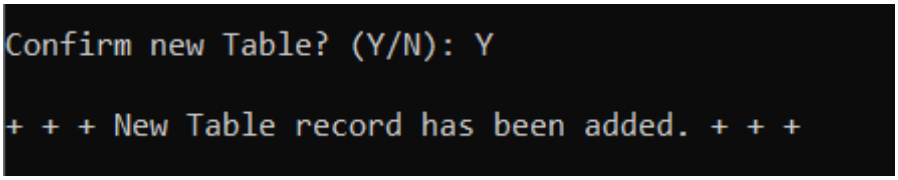


Figure 2.2: User inputs ‘Y’

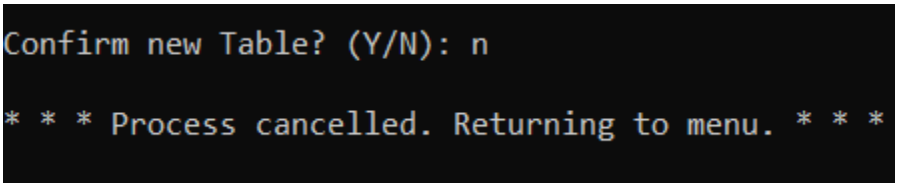


Figure 2.3: User inputs ‘N’

The system repeats this operation every time the user uses the “Add Record” function. When the maximum record limit, which is 100 records, is reached, the system will notify the user that the maximum record limit has been reached. If the user attempts to add more records after the maximum record limit has been reached, the system prohibits the user from entering more records and will not prompt the user for any input, then the user is immediately redirected to the main menu.

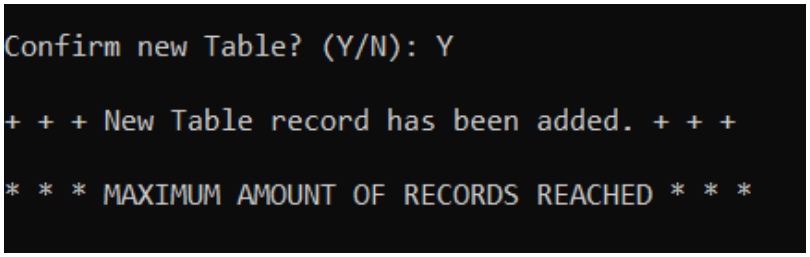


Figure 2.4: System notifies the user that the maximum amount of records is reached

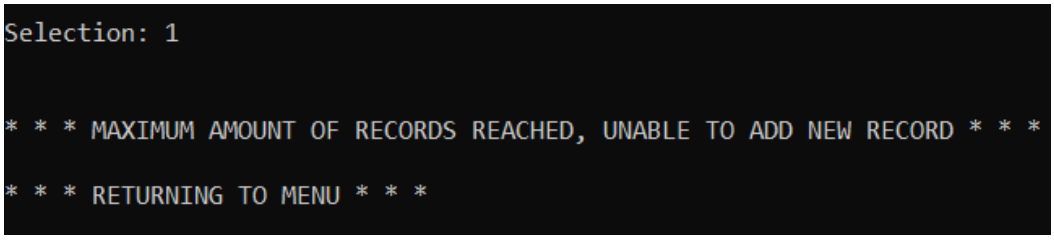


Figure 2.5: System prohibits the user from entering a new record once the limit is reached

- **Edit Record**

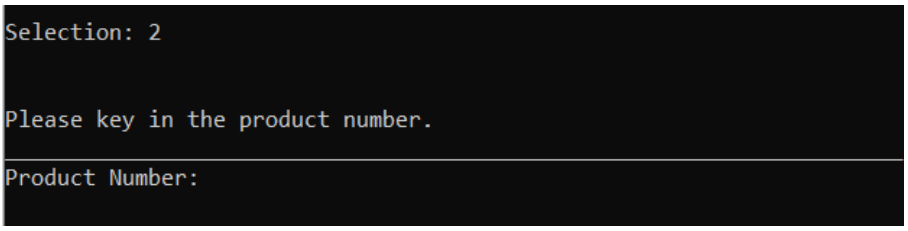


Figure 3.0: User inputs 2, system redirects user to “Edit record” function.

If the user inputs “2” in the menu, the system redirects the user to the “Edit record” function. The purpose of the “Edit record” function is mainly to allow the user to edit the values of width, height, and length for a table record. In this function, the system first prompts the user for the product number of the table record that they want to edit. If the user inputs a product number that does not exist in the system, the system will notify the user that there is no existing record of the product number that they inputted, and it returns them back to the main menu.

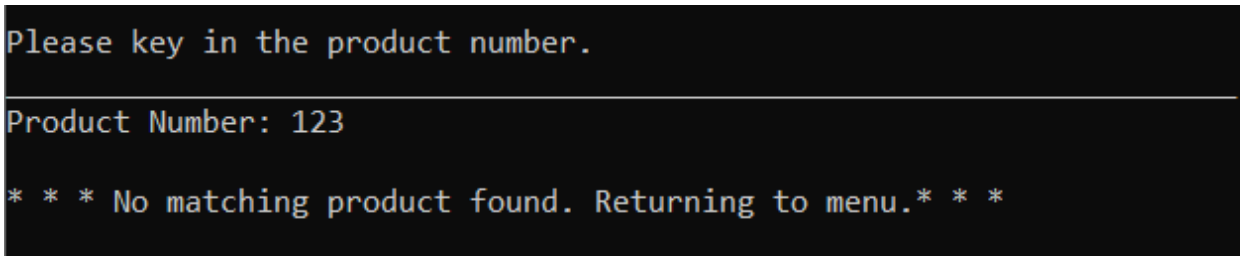


Figure 3.1: Entered product number does not exist in system, user is redirected to main menu.

However, if a matching table record is found, the system will display the product’s current details to the user. The system then asks the user to key in the new values of the length, width, and height for the product. The system will notify the user of any invalid inputs which are any non-numeric values, and continuously prompts the user until a valid input is keyed in.

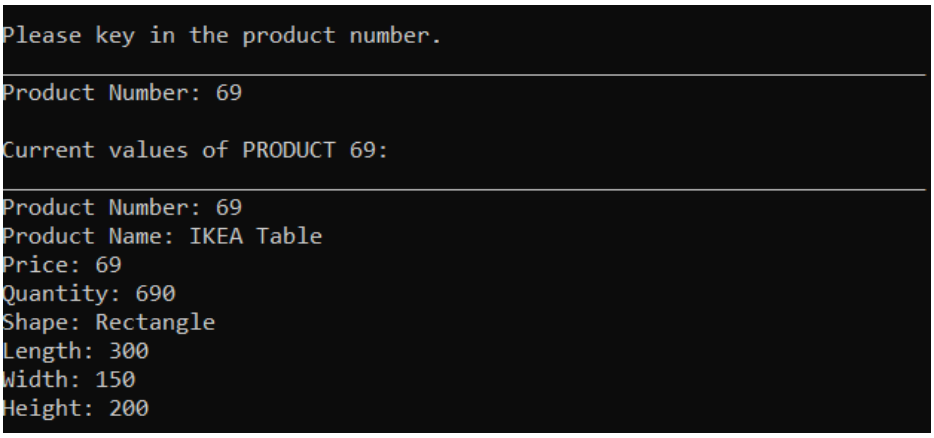


Figure 3.2: User enters valid product number; details of product are displayed.

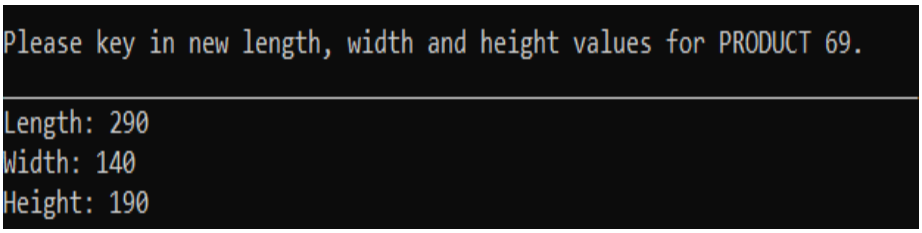


Figure 3.3: Process of entering new dimension values.

Once the user finishes inputting the new dimension values for the product, the system asks the user to confirm if they want to change the old values to the new ones. If the user enters 'Y', the system will print out a message saying that the new changes have been applied to the table record successfully. Conversely, if the user enters 'N', the system will discard the changes made to the table record and notify the user that the edit process is cancelled. Regardless of the choice, the user will be redirected back to the main menu where they are prompted another selection input.

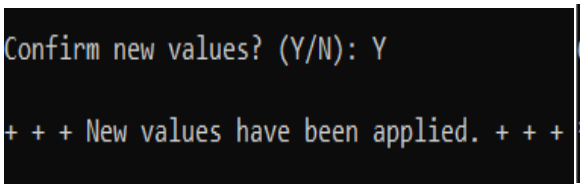


Figure 3.4: User inputs 'Y'

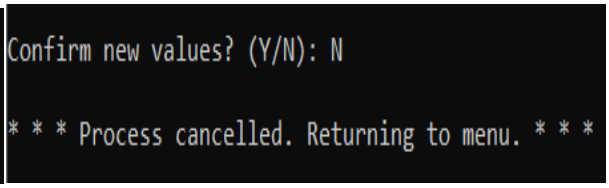


Figure 3.5: User inputs 'N'

- **Search Record**

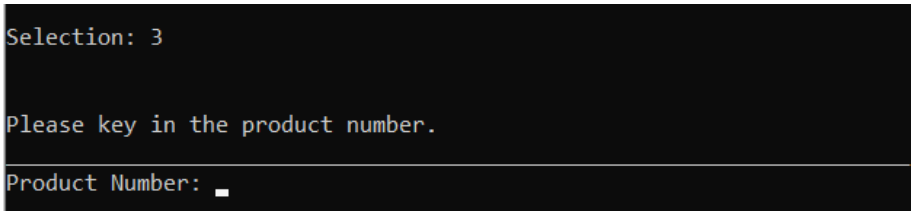


Figure 4.0: User inputs 3, system redirects user to "Search record" function.

If the user inputs "3" in the menu, the system redirects the user to the "Search record" function. The purpose of the "Search record" function is to allow the user to search for a specific table record and to check its details. In this function, the system prompts the user to key in the product number of the table record that they want to search for. If the user enters a product number that does not exist in the system, the system will notify the user that there is no existing record of the product number that they entered, and it returns them back to the main menu.

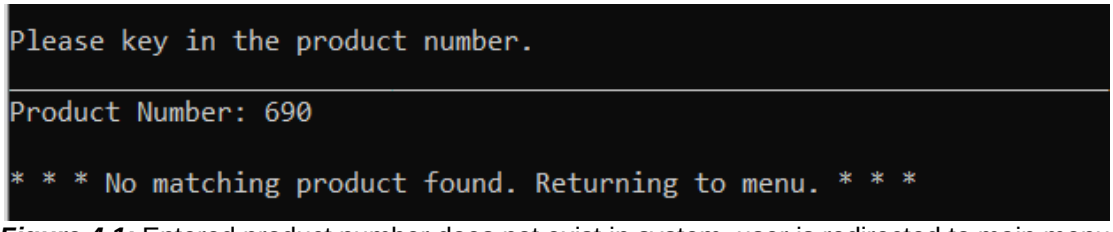


Figure 4.1: Entered product number does not exist in system, user is redirected to main menu.

However, if a matching table record is found, the system will display the product's details to the user. After that, the process immediately resolves itself to complete and the user is redirected to the main menu.

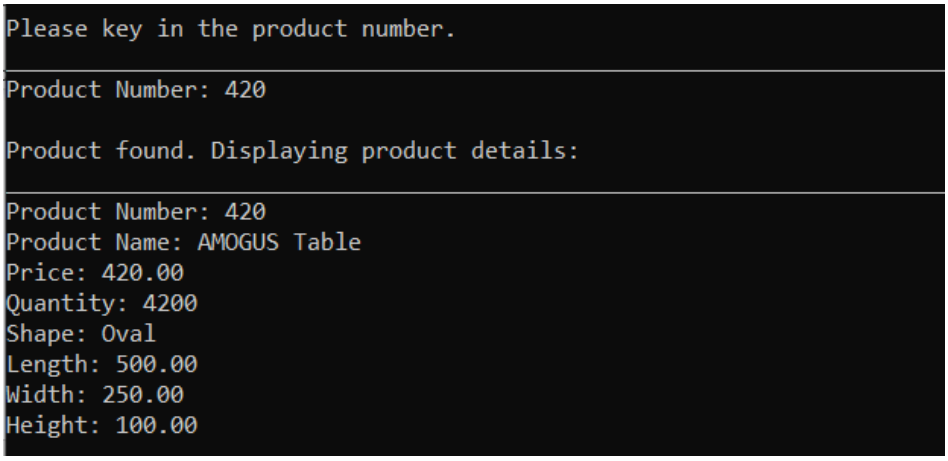
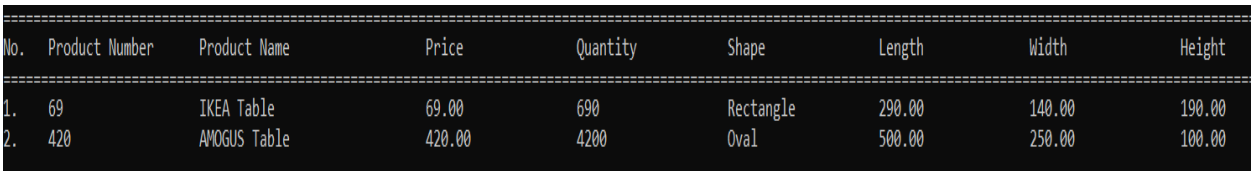


Figure 4.2: Matching product number found; details of product are displayed.

- **Check Records**

If the user inputs "4" in the menu, the system redirects the user to the "Check records" function. The purpose of the "Check records" function is to allow the user to check the details of all existing table records in the system. In this function, the system displays each product out to the user by row, and each type of product details are put under their respective columns. The table records are listed from top to bottom based by their time of creation. The price, length, width, and height values of the products are all set to 2 decimal places for precision. After that, the process immediately resolves itself to complete and the user is returned to the main menu. If the user had not added any table record into the system yet, the system would show "NO RECORDS" to the user.



No.	Product Number	Product Name	Price	Quantity	Shape	Length	Width	Height
1.	69	IKEA Table	69.00	690	Rectangle	290.00	140.00	190.00
2.	420	AMOGUS Table	420.00	4200	Oval	500.00	250.00	100.00

Figure 5.0: System displays a table containing all existing table records.

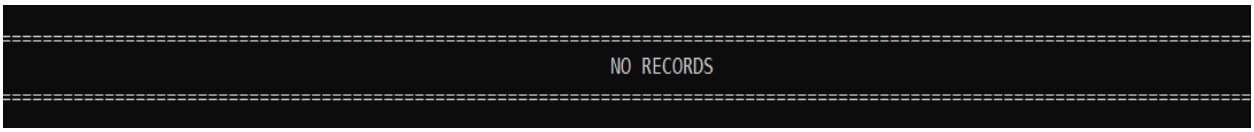


Figure 5.1: System displays "NO RECORDS" if there is no existing table record.

- **Exit**

Figure 6.0: User inputs 6, system redirects user to “Exit” function

Lastly, if the user inputs “5” in the menu, the system redirects the user to the “Exit” function. This function can be used by the user to end and quit the system. As seen in ***Figure 6.0***, the system displays a message and prompts the user for confirmation to quit the system or not. If the user inputs ‘Y’, the system ends; if the user inputs ‘N’, the system does not end, and it redirects the user back to the main menu.

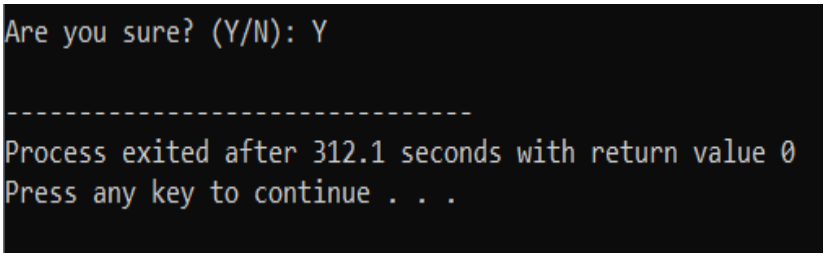


Figure 6.1: User inputs ‘Y’; system ends

Figure 6.2: User inputs ‘N’; user is redirected to main menu

Bibliography List

cplusplus.com (2020) *istream::ignore*. cplusplus.com. Available from http://www.cplusplus.com/reference/istream/istream/ignore/ [accessed 26 March 2021].

cplusplus.com (2020) <*iomanip*>. cplusplus.com. Available from https://www.cplusplus.com/reference/iomanip/ [accessed 26 March 2021].

cplusplus.com(2020) *toupper.* cplusplus.com. Available from http://www.cplusplus.com/reference/cctype/toupper/ [accessed 26 March 2021].

Malik, D.S. (2008) *Introduction to C++ Programming: Brief Edition*. Massachusetts, USA: Cengage Course Technology.

Visual Paradigm (2020) *What is Class Diagram?*. Visual Paradigm. Available from https://www.visual-paradigm.com/guide/uml-unified-modeling-language/what-is-class-diagram/ [accessed 26 March 2021].