| ***Computer Engineering Department*** |
| --- |
| ***CE100L: Computing Fundamentals & Programming*** |

| ***Course Instructor: Usama Bin Shakeel*** | ***Dated: 21/01/2022*** |
| --- | --- |
| ***Teaching Assistant: Aqsa Khalid*** | ***Semester: Fall 2021*** |
| ***Lab Engineer: Nadir Abbas*** | ***Batch: BSCE2021*** |

# **Assignment 13. Challenging Problem Practice in C++**

| **Name** | **Roll number** | **Report**  **(out of 100)** | **Scaled to 10** | **Total**  **(out of 10)** |
| --- | --- | --- | --- | --- |
| Muhammad Abubakar Saif | BSCE21017 |  |  |  |

Checked on: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Signature: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

## **Objective**

The objective of this lab is to understand open ended problem solving.

## **Equipment and Component**

| **Component Description** | **Value** | **Quantity** |
| --- | --- | --- |
| Computer | Available at home | 1 |

## **Conduct of Assignment**

1. Students are required to perform assignment individually.
2. In case the lab experiment is not understood, the students are advised to seek help from the course instructor, lab engineers, assigned teaching assistants (TA) and lab attendants.

## **Theory and Background**

A challenge is something new and difficult which requires great effort and determination.Every problem has a solution, we just need to think and try to solve it. We can solve every problem, we just need some courage to solve it. It depends on the problem, how much courage we require to solve that problem.



Figure 1: \*What is Open Ended Problem Solving??

\*https://medium.com/@aitisam.zafeer88/problem-solving-challenge-322a3ee7db23

**Assignment Task**

1. A customer wants you to design and then build an online store. While interviewing the customer following information was extracted:

Track products and related information.   
To do: extract details (i.e. other information to track) from interview notes.

Abilities the customer wants in the system:

Identify a product that will be for sale. You should read in the name, model number, wholesale cost, and retail cost for the product.Enter a new customer. You should read in the customer name and ID.

Take a shipment of new products. Read in the model number and quantity. If you don't know what the product is that you're getting, reject the shipment, otherwise add that to inventory.

Let a customer buy something. The customer ID, product model number, and quantity should be taken as input. If there is sufficient quantity of the product on hand, then the customer should be charged that amount and the product be deducted from inventory. If there is not sufficient quantity, the sale should be rejected. Let a customer make a payment. Read in the customer ID and the amount of payment. It's OK for customers to have a positive balance, but they cannot make negative payments. Find out about a customer: enter a customer ID number, and print out the customer's name, current balance, and a list of what the customer has previously purchased. Find out about a product: enter a model number and get the name of the product, the amount that has already been sold, and the amount in inventory. Print lists of all information about all customers and all products.

| **int** number = 0;  **void** productEntry() { *//used by store owner to enter products in inventory which is available in store*  cin.ignore();  string user; *// or being ordered but not received*  ofstream write;  write.open(**"Inventory.txt"**, ios::app); *//will open "Inventory" file in append mode*  product add; *//declare structure variable "add"*  again: *//goto label*  cout << **"Enter the product name: "** << endl;  cin.getline(add.name, NUM); *//capable of taking input with spaces from user*  spaceRemover(add.name); *//replaces space in given char array by user with "\_"*  write << add.name << **" "**; *//write product name in file*  cout << **"Enter its model/type: "** << endl;  cin >> add.modelNum;  write << add.modelNum << **" "**;  cout << **"Enter its wholesale price: "** << endl;  cin >> add.wholesale;  write << add.wholesale << **" "**;  cout << **"Enter its retail price: "** << endl;  cin >> add.retail;  write << add.retail << **" "**;  cout << **"Enter the quantity in your stock: (Enter 0 if it is out of stock/hasn't received yet)"** << endl;  cin >> add.quantity;  write << add.quantity << **" "**;  write << endl;  cout << **"Do you want to enter next product: (Y/N)"**;  *// cin >> user;*  **if** (user == **"Y"** || user == **"y"** || user == **"Yes"** || user == **"yes"** || user == **"yeS"**) {  cin.ignore();  **goto** again;  }  **return**;  }  **void** shoppingPortal(string ID) {  cout << **"Your User ID is : "** << ID << endl;  product user;  inventoryList(1);  cin.ignore();  cout << **"Enter the product you want to buy: "**;  cin.getline(user.name, NUM);  spaceRemover(user.name);  **return**;  }  **void** spaceRemover(**char** test[NUM]) {  **for** (**int** i = 0; i < NUM; ++i) {  **if** (test[i] == **' '**) {  test[i] = **'\_'**;  }  }  }  **void** productCount(**int** &count) {  ifstream read;  read.open(**"Inventory.txt"**);  **char** line[NUM];  **while** (!read.eof()) {  read.getline(line, NUM);  *//cout << line << endl;*  count++;  }  *//cout << count << ";" << endl;*  **return**;  }  **void** inventoryList(**int** pointer) { *//pointer = 0 for shipment receiver, pointer = 1 for for displaying inventory*  fstream store;  store.open(**"Inventory.txt"**, ios::in);  **int** i = -1;  productCount(number); *//will store "number" of products stored in inventory*  product \*local = **new** product[number];  cin.ignore();  *//for (int i = 0; ; ++i) {*  **do** {  i++;  store >> i >> local[i].name >> local[i].modelNum >> local[i].wholesale >> local[i].retail  >> local[i].quantity;  **if** (pointer == 1) {  *//cin.ignore();*  cout << local[i].name << local[i].modelNum << local[i].wholesale << local[i].retail << local[i].quantity;  cout << endl;  }  } **while** (i < number && local[i].wholesale != 0);  **if** (pointer == 0) {  shipmentEntry(local);  }  *// delete[] local;*  **return**;  }  **void** shipmentEntry(product local[]) {  cin.ignore();  string model;  **char** label[NUM];  **int** amount;  cout << **"Enter Product: "**;  cin.getline(label, NUM);  spaceRemover(label);  cout << **"Enter Product model: "**;  cin >> model;  *//cout<<model;*  *//cout<<label[7];*  **for** (**int** i = 0; i < number; ++i) {  *//cout<<i<<" "<<local[i].modelNum<<local[i].name<<endl;*  **if** (local[i].modelNum == model || local[i].name == label) {  cout << **"Enter shipped quantity of product: "**;  cin >> amount;  local[i].quantity += amount;  updateInventory(local);  }  }  }  **void** updateInventory(product latest[]) {  ofstream refresh;  refresh.open(**"Inventory.txt"**);  **for** (**int** i = 0; i < number - 1; ++i) {  refresh << latest[i].name << **" "**;  refresh << latest[i].modelNum << **" "**;  refresh << latest[i].wholesale << **" "**;  refresh << latest[i].retail << **" "**;  refresh << latest[i].quantity << **" "**;  refresh << endl;  }  }  **void** customerLogin() {  string username;  string ID;  again:  system(**"clear"**);  cout << **" \*\*\*\*\*\*\*\*\* ~ WELCOME TO INDUS ONLINE STORE ~ \*\*\*\*\*\*\*\*\*\* "** << endl;  cout << endl;  cout << **" \*\*\*\*\*\*\*\* Customer Login Portal \*\*\*\*\*\*\*\*\* "** << endl;  cout << **" \*\*\*\*\*\*\*\* Press 1 to login \*\*\*\*\*\*\*\*\* "** << endl;  cout << **" \*\*\*\*\*\*\*\* Don't worry! if you don't have account, Press 2 to make account \*\*\*\*\*\*\*\*\* "** << endl;  **int** option;  cin >> option;  **if** (option == 1) {  cout << **" YOU HAVE AN EXISTING ACCOUNT "** << endl;  user:  cout << **" ENTER YOUR USERNAME: "**;  cin >> username;  pass:  cin.ignore();  cout << **" ENTER YOUR ID : "**;  cin >> ID;  **bool** check = check\_account(username, ID);  **if** (check == **true**) {  **bool** check\_pass = check\_ID(username, ID);  **if** (check\_pass == **true**) {  shoppingPortal(ID);  *// productEntry();*  *// cout << "Write Main Customer Function Here"<<endl;*  } **else** {  cout << **"\*\*\*\* ID is NOT valid \*\*\*\*"** << endl;  **goto** pass;  }  } **else** {  cout << **"\*\*\*\* Your USERNAME does NOT exists \*\*\*\*"** << endl;  **goto** user;  }  } **else if** (option == 2) {  {  cout << **"\*\*\*\* YOU WILL CREATE AN ACCOUNT \*\*\*\* "** << endl;  cout << **" SET A USERNAME: "**;  cin >> username;  cout << **" SET A ID: "**;  cin >> ID;  account\_creation(username, ID);  cout << endl;  cout << **"\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*"** << endl;  cout << **"\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*YOUR ACCOUNT HAS BEEN CREATED\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*"** << endl;  cout << **"\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*"** << endl;  cout << **"\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*LOGIN TO PROCEED NEXT\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*"** << endl;  **goto** again;  }  } **else** {  cout << **"Wrong choice"**;  }  **return**;  }  **struct** user\_acc {  string user;  string identity;  };  fstream credentials;  **bool** check(string username) {  ifstream fopen;  string line;  fopen.open(**"credentials.txt"**);  **while** (getline(fopen, line)) {  **int** pos = line.find(**":"**);  **int** End = line.find(**","**);  string sub = line.substr(pos + 1, End - pos - 1);  **if** (sub == username) {  fopen.close();  **return true**;  }  }  fopen.close();  **return false**;  }  **void** account\_creation(string username, string ID) {  string line;  user\_acc creds;  ofstream fwrite;  fwrite.open(**"credentials.txt"**, std::ios\_base::app);  **while** (check(username)) {  *//cout << "Yes String is there" << endl;*  *//cout << sub << endl;*  cout << **"The username you entered exists, Kindly change your username"** << endl;  cout << **"Enter username: "**;  cin >> username;  }  creds.user = username;  creds.identity = ID;  *//username:maryam, ID : shahid*  fwrite << **"username:"** + creds.user + **",ID:"** + creds.identity + **"\n"**;  fwrite.close();  }  **bool** check\_account(string username, string ID) {  string line;  ifstream fopen;  fopen.open(**"credentials.txt"**);  **while** (getline(fopen, line)) {  **int** pos = line.find(**":"**);  **int** End = line.find(**","**, pos);  string sub = line.substr(pos + 1, End - pos - 1);  **if** (sub == username) {  *//cout << "Yes String is there" << endl;*  *//cout << sub << endl;*  **return true**;  }  }  fopen.close();  **return false**;  }  **bool** check\_ID(string username, string ID) {  string line;  ifstream fopen;  fopen.open(**"credentials.txt"**);  **while** (getline(fopen, line)) {  **int** pos = line.find(**":"**);  **int** end = line.find(**","**, pos);  string sub = line.substr(pos, end - pos);  **if** (sub == **":"** + username) {  *//cout << "Yes String is there" << endl;*  *//cout << sub << endl;*  *//return true;*  pos = line.find(**","**);  end = line.find(**":"**, pos);  **int** end1 = line.find(**":"**, end);  string sub1 = line.substr(end, end1);  **if** (sub1 == **":"** + ID) {  *//cout << "ID is " << sub1 << endl;*  **return true**;  }  }  }  **return false**;  } |
| --- |

2. Write a method to replace all spaces in a string with '%2e: You may assume that the string has sufficient space at the end to hold the additional characters, and that you are given the "true" length of the string. (Note: In C++, please use a character array so that you can perform this operation in place.)

EXAMPLE

Input: "Mr John Smith JJ”, 13

Output: "Mr%2eJohn%2eSmith"

| void spaceReplace(char test[NUM]) { //Not completed, working  for (int i = 0; i < NUM; ++i) {  if (test[i] == ' ') {  test[i] = '\_';  }  }  cout<<test<<endl;  } |
| --- |

#### **Assessment Rubric for Lab**

**Method for assessment:**

Lab reports and instructor observation during lab sessions. Outcome assessed:

a. Ability to conduct experiments, as well as to analyze and interpret data (P) b. Ability to function on multi-disciplinary teams (A)

c. Ability to use the techniques, skills, and modern engineering tools necessary for engineering practice (P)

| Performance metric | Mapping (task no. and description) | | Max marks | Exceeds expectation | Meets expectation | Does not meet expectation | Obtained marks |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 1. Realization of experiment (a) | 1 | Functionality | 40 | Executes without errors excellent user prompts, good use of symbols, spacing in output. Through testing has been completed (35-40) | Executes without errors, user prompts are understandable, minimum use of symbols or spacing in output. Some testing has been completed (20-34) | Does not execute due to syntax errors, runtime errors, user prompts are misleading or non-existent. No testing has been completed (0-19) |  |
| 2. Teamwork (b) | 1 | Group Performance | 5 | Actively engages and cooperates with other group member(s) in effective manner (4-5) | Cooperates with other group member(s) in a reasonable manner but conduct can be improved (2-3) | Distracts or discourages other group members from conducting the experiment (0-1) |  |
| 3. Conducting experiment (a, c) | 1 | On Spot Changes | 10 | Able to make changes (8-10) | Partially able to make changes (5-7) | Unable to make changes (0-4) |  |
| 2 | Viva | 10 | Answered all questions (8-10) | Few incorrect answers (5-7) | Unable to answer all questions (0-4) |  |
| 4. Laboratory safety and disciplinary rules (a) | 1 | Code commenting | 5 | Observes lab safety rules; handles the equipment and parts with care and adheres to the lab disciplinary guidelines aptly (4-5) | Generally observes safety rules and disciplinary guidelines with minor lapses (2-3) | Disregards lab safety and disciplinary rules (0-1) |  |
| 5. Data collection (c) | 1 | Code Structure | 5 | Excellent use of white space, creatively organized work, excellent use of variables and constants, correct identifiers for constants, No line-wrap (4-5) | Includes name, and assignment, white space makes the program fairly easy to read. Title, organized work, good use of variables (2-3) | Poor use of white space (indentation, blank lines) making code hard to read, disorganized and messy (0-1) |  |
| 6. Data analysis (a, c) | 1 | Algorithm | 20 | Solution is efficient, easy to understand, and maintain (15-20) | A logical solution that is easy to follow but it is not the most efficient (6-14) | A difficult and inefficient solution (0-5) |  |
| 7. Computer use (c) | 1 | Documentation | 5 | Timely documented (4-5) | Late documented (2-3) | Not documented (0-1) |  |
|  | Max Marks (total): | | 100 | Obtained Marks (total): | | |  |

Lab Engineer Signature: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_