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

| ***Course Instructor: Usama Bin Shakeel*** | ***Dated: 10/01/2022*** |
| --- | --- |
| ***Teaching Assistant: Aqsa Khalid*** | ***Semester: Fall 2021*** |
| ***Lab Engineer: Nadir Abbas*** | ***Batch: BSCE2021*** |

# **Lab 12A. Open Ended Problem Solving in C++**

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

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

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

## **Objective**

The objective of this lab is to understand the complexity level in arrays, recursion, and arrays.

## **Equipment and Component**

| **Component Description** | **Value** | **Quantity** |
| --- | --- | --- |
| Computer | Available in lab | 1 |

## **Conduct of Lab**

1. Students are required to perform this experiment 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**

**Open-ended problem** is a problem that has several or many correct answers, and several ways to the correct answer(s). The Open-Ended Approach provides students with "experience in finding something new in the process"(Shimada 1997). It is basically facilitating the development of creative problem solving skills.

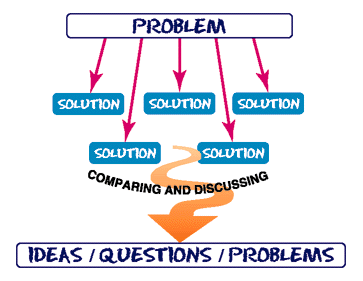


Figure 1: \*What is Open Ended Problem Solving??

**Lab Task**

1. A small airline has just purchased a computer for its new automated reservations system. You’ve been asked to program the new system. You are to write a program to assign seats on each flight of the airline’s only plane (capacity: 10 seats).

\*https://mste.illinois.edu/users/aki/open\_ended/WhatIsOpen-ended.html

Your program should display the following menu of alternatives—Please type 1 for "First Class" and Please type 2 for "Economy". If the person types 1, your program should assign a seat in the first class section (seats 1– 5).If the person types 2, your program should assign a seat in the economy section (seats 6–10). Your program should print a boarding pass indicating the person’s seat number and whether it’s in the first class or economy section of the plane.

Your program should, of course, never assign a seat that has already been assigned. When the first class section is full, your program should ask the person if it’s acceptable to be placed in the economy section (and vice versa).If yes, then make the appropriate seat assignment. If no, then print the message "Next flight leaves in 3 hours."

| void bookingFlights(){  int seata;  int seatb;  int seatc; //initializing  char card[200];  int opt;  int opta;  fstream myfile;  fstream myfilea; //files  char Class;  cout<<"please tell me your name = ";  cin.getline(card,200); //taking names  cout<<endl;  cout<<"please enter your seat number = "; //seat number  cout<<seatc<<endl;  cout<<"please enter your class "; //chosing class  if (Class=='A'){  cout<<"1st class";  }  else if (Class =='B'){  cout<<"Economy class";  }  do{  cout<<"please enter 1 for First Class"<<endl;  cout<<"please enter 2 for Economy class"<<endl;  cin>>opt;  switch(opt){  case 1:  {  myfile.open("seata");  myfile>>seata;  if(seata<5){  seata++;  myfile.close();  Class=='A';  cout<<"you are assigned an first class ticket plz enjoy yourself."<<endl;  seatc=seata;  }  else if (seatb<=5){  myfilea.open("seatb");  myfilea>>seatb;  if(seatb<5){  cout<<"1.would u like to go from economy class?"<<endl;  cout<<"2.or would u like to wait for the next flight?"<<endl;  cin>>opta;  if(opta==1){  seatb++;  myfilea.close();  Class=='B';  cout<<"you are assigned an economy class ticket plz enjoy yourself.";  seatc=seatb;  }  }  else{  cout<<"next flight will take off in 3 hours,you can wait."<<endl;  break;  }  }  }  case 2:  {  myfilea.open("seatb");  myfilea>>seatb;  if(seatb>=5){  seatb++;  Class='B';  cout<<"you are given an economy class ticket plz enjoy.";  seatc=seatb;  }  break;  }  }  }  while(opt>=1 && opt<=2);  }  void changingFile(){  int seata;  int seatb;  fstream file;  fstream filea;  file.open("seata");  filea.open("seatb");  ofstream temp("temp");  ofstream tempa("tempa");  temp<<seata;  tempa<<seatb;  file.close();  filea.close();  remove("seata");  rename("temp", "seata");  remove("seatb");  rename("temp", "seatb");  } |
| --- |

1. Tomorrow is a holiday on the 14th of august (day of independence of pakistan) . You are planning to go to Joyland but this park is at a considerable distance so you are going to check ticket criteria online .. where you will enter the number of persons going, hours you will stay there and swing you gonna choose furthermore

#### ——> the rates for discovery

#### \*children below 10 are not allowed to sit on discovery

#### \*10-15(age) wins 10 % discount

#### 15-20 wins 5% discount

#### \*while no for above 20

#### ——> the rates for maryland swing

#### \* 1-5 (age) wins 50 % discount

#### \* 5-10 25% discount

#### \* while no for above

#### and program will then ask to enter age of person and will run until you enter the age of last family member program will then tell you the amount due after each person’s age you enter parking fee which is 10/hour and price of ticket is PKR 100

| void joyLandDiscovery() {  int members; //initialization  int age = 1; //initialization  float discount; //initialization  int priceOfTicket = 100; //initialization  float price; //initialization  cout << "please enter how many members of family are u going to take a ride of discovery= ";  cin >> members; //asked for members of family  for (int i = 0; i < members; i++) { //a loop for member of family  cout << "please enter the age of member = "; //asked age  cin >> age;  if (age < 10) {  cout << "we are really sorry ,but you are not allowed to sit on discovery -\_-" << endl; //if age if less than 10 then they cant take this ride we apply condition  cout << endl;  } else if (age >= 10 && age <= 15) { //condition that if age is between 10 to 15  discount = priceOfTicket \* (0.1); //then a discount is given  cout << "you are given an discount of 10% , that is = " << discount << endl; //displaying  price = priceOfTicket - discount;  cout << "now the price if ride is = " << price << endl; //price after discount  cout << endl;  } else if (age >= 16 && age <= 20) { //condition that if age is between 16 to 20  discount = priceOfTicket \* (0.05); //then a discount is given  cout << "you are given an discount of 5% , that is = " << discount << endl; //displaying  price = priceOfTicket - discount;  cout << "now the price if ride is = " << price << endl; //price after discount  cout << endl;  } else {  cout << "you can sit on ride, but you are not given any discount -\_-" << endl; //more than 20 do not get any discount  cout << endl;  }  }  }  void joyLandMaryLand() {  int age = 1; //initialization  int priceOfTicket = 100;  float discount;  int members;  float price;  cout << "please enter how many members of family are u going to take a ride of maryLand = "; //asking members  cin >> members;  for (int i = 0; i < members; i++) { //a loop for family members  cout << "please enter the age of members = ";  cin >> age;  if (age <= 5) {  discount = priceOfTicket \* (0.5); //discount formula  cout << "you are given an discount of 50%, that is = " << discount << endl; //displaying  price = priceOfTicket - discount; //finding price  cout << "now the price if ride is = " << price << endl; //displaying  cout << endl;  } else if (age > 5 && age <= 10) { // condition that if age is less than 10  discount = priceOfTicket \* (0.25); //discount formula  cout << "you are given an discount of 25% , that is =" << discount << endl;  price = priceOfTicket - discount; //price  cout << "now the price if ride is = " << price << endl; //displaying price  cout << endl;  } else {  cout << " you can take ride , but you are not given any discount -\_-" << endl;  cout << endl;  }  }  }  void priceOfParking() {  int hours;  int PRICEOfPARKING;  cout << "for how many hours will u guys stay ?" << endl; //asking hours  cin >> hours;  PRICEOfPARKING = hours \* 100; //priceOfParking formula  cout << " the Price of parking is = " << PRICEOfPARKING << endl; //displaying price  cout << endl;  } |
| --- |

1. A company pays its salespeople on a commission basis. The salespeople each receive $200 per week plus 9 percent of their gross sales for that week. For example, a salesperson who grosses $5000 in sales in a week receives $200 plus 9 percent of $5000, or a total of $650.

Write a program (using an array of counters) that determines how many of the salespeople earned salaries in each of the following ranges.

a) $200–299 b) $300–399 c) $400–499 d) $500–599 e) $600–699 f) $700–799 g) $800–899 h) $900–999 i) $1000 and over.

Write the program or 20 salespersons.

| void salaries()  {  double gross; //initializing  double \*arr = new double [20]; //giving dynamic memory ellocation  double grosses; //initializing  double \*arr2 = new double [9]; //giving dynamic memory ellocation  arr2[0]=0;  arr2[1]=0; //giving value zero at indexes 0 to 8  arr2[2]=0;  arr2[3]=0;  arr2[4]=0;  arr2[5]=0;  arr2[6]=0;  arr2[7]=0;  arr2[8]=0;  for(int i=0;i<20;i++){ //loop for people's gross  cout<<"please enter the gross value = "; //getting gross value from user  cin>>arr[i]; //storing that value in an array  grosses=200+ ((0.09)\*arr[i]); //adding that 200 in the array  cout<<"grosses of person is = "<<grosses<<endl; //displaying gross  if(grosses>=200 && grosses<=299) //condition for the gross to lie in this range  {  arr2[0]++; //storing how many people lie in this range  }  else if(grosses>=300 && grosses<=399){ //condition for the gross to lie in this range  arr2[1]++; //storing how many people lie in this range  }  else if(grosses>=400 && grosses<=499){ //condition for the gross to lie in this range  arr2[2]++; //storing how many people lie in this range  }  else if(grosses>=500 && grosses<=599){ //condition for the gross to lie in this range  arr2[3]++; //storing how many people lie in this range  }  else if(grosses>=600 && grosses<=699){ //condition for the gross to lie in this range  arr2[4]++; //storing how many people lie in this range  }  else if(grosses>=700 && grosses<=799){ //condition for the gross to lie in this range  arr2[5]++; //storing how many people lie in this range  }  else if(grosses>=800 && grosses<=899){ //condition for the gross to lie in this range  arr2[6]++; //storing how many people lie in this range  }  else if(grosses>=900 && grosses<=999){ //condition for the gross to lie in this range  arr2[7]++; //storing how many people lie in this range  }  else  {  arr2[8]++; //storing how many people lie in this range  }  }  cout<<"the persons who lie in the range of 200$ to 299$ are:"<<arr2[0]<<endl;  cout<<"the persons who lie in the range of 300$ to 399$ are:"<<arr2[1]<<endl;  cout<<"the persons who lie in the range of 400$ to 499$ are:"<<arr2[2]<<endl; //displaying the poeople who lie in this range  cout<<"the persons who lie in the range of 500$ to 599$ are:"<<arr2[3]<<endl;  cout<<"the persons who lie in the range of 600$ to 699$ are:"<<arr2[4]<<endl;  cout<<"the persons who lie in the range of 700$ to 799$ are:"<<arr2[5]<<endl;  cout<<"the persons who lie in the range of 800$ to 899$ are:"<<arr2[6]<<endl;  cout<<"the persons who lie in the range of 900$ to 999$ are:"<<arr2[7]<<endl;  cout<<"the persons who lie in the range of 1000 and above are:"<<arr2[8]<<endl;  }  OUTPUT |
| --- |

#### **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: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_