Yaseen Khan
Ethan Lew
Nicholas Lozano

Team AYEN Big-Oh Analysis

**Function #1: CreateRoute() - O(n)**
Description: Finds the next closest campus from the previous campus recursively.

Code:

```
void tripRoutePlanner::createRoute(QString campus)
{
  // add the campus to the route queue
  route.push(campus);  + 1 (inserts campus on queue always runs in O(1))

  // remove the campus from the list of campuses that still need to be visited
  auto it = std::find(campusesToVisit.begin(), campusesToVisit.end(), campus);  + n (find runs
in O(n) time for STL list) + 1 (Assigning iterator)
  if (it != campusesToVisit.end())    + 1 (Selection Statement)
  {
     campusesToVisit.erase(it);     +1 (erases campus stored at iterator)
  }

  // base case, no more campuses to visit in this route
  if (campusesToVisit.size() <= 0)   +1 - Selection statement
  {
     // assign this campus to finalCampus and exit function
     finalCampus = campus;    +1 - assigns a variable to another variable
     return;                  + 1  - jumps out of function
  }

  else
  {
     // initialize variables to hold closest campus
     QString closestCampus = campusesToVisit.front();  +1 (Gets First element of list)
     double shortestDist = database->getDistance( campus, campusesToVisit.front() );   + 9
(getDistance runs in O(1) time) + 1 (Assignment)
     auto it = campusesToVisit.begin();    +1 (Gets the iterator of first element)

   -    Total Instruction Time: 12n + 1
     while (it != campusesToVisit.end())  + n (amount of campuses) + 1 (loop overhead)
     {
        // update the closest campus if the currently accessed campus in the vector is closer than
       // previous closest campus
        if (database->getDistance(campus, *it) < shortestDist)    + 1 (Selection)
```

```
        {
            closestCampus = *it;                          + 1 (Assigning variables)
            shortestDist = database->getDistance( campus, *it );    + 9 (getDistance runs in O(1)
time) + 1 (Assigning variables)
        }

        // increment iterator
        it++;                                  + 1 (Increments Iterator)
    }

    // finally add to the total distance counter
    totalDistance += shortestDist;     + 1 (Adds to total Distance)

    // recursive call
    createRoute(closestCampus);    + n (recursively goes through all campuses in Route)
  }
}
```

**Total Running Time for CreateRoute(): O(n): 1 + (n + 1) + 1 + 1 + 1 + 1 + 1+ 1 + 10 + 1 + (12n + 1) + 1 + n = 14n + 21**

**Function #2: Add Campus() - O(n)**

Description: Allows the administrator to add campuses to a database through a text file

Code:

```
void addcampuses::on_addFile_clicked()
{
  string startingDist;     + 1 (Declaring variable)
  string endingDist;     + 1 (Declaring variable)
  double distance;     + 1 (Declaring variable)
  int idNum;     + 1 (Declaring variable)
  bool success;     + 1 (Declaring variable)

  // Opens up file on computer
  QString fileName = QFileDialog::getOpenFileName(this, tr("Open File"), "/Downloads", tr("Txt
Files (*.txt)"));     + 1 (Allows for text files)
  QFile file(fileName);     + 1 (Reads fileName)

  std::ifstream inFile;     + 1 (Declaring variable)
  inFile.open(fileName.toStdString());   + 1 (Declaring variable)

  if (!file.open(QIODevice::ReadOnly | QIODevice::Text))   + 1 (Selection) - else is bigger
```

```
  {
     QMessageBox::information(this, QObject::tr("System Message"), tr("File cannot be found.
Add a different file"));
  }

  else
  {
     Total Running Time: 21n + 1(loop overhead)
     while (inFile)  + n (number of inputs) + 1 (loop overhead)
     {
        // reads the two campuses and distances
        getline(inFile, startingDist);       + 1 (reads campus)
        getline(inFile, endingDist);       + 1 (reads campus)
        inFile >> distance;                + 1 (reads distance)
        inFile.ignore(10000, '\n');        + 1 (ignores punctuation)

        // gets the starting and ending campus
        QString startDist = QString::fromStdString(startingDist);   + 1 (Assigns variable)
        QString endDist = QString::fromStdString(endingDist);    + 1 (Assigns variable)

        // Checks if Campus is added
        if (campusExists(startDist, endDist, distance))   + 1 (Selection) - else is bigger
        {
           QMessageBox::information(this, QObject::tr("System Message"), tr("Campus already
exists. Cannot add campus"));
           break;
        }
        else
        {
           QSqlQuery query;    + 1 (Declaring variable)
           query.prepare("SELECT max(ID) from CAMPUSES"); // get the maximum id from the
                                                          // table    + 1 (Gets ID)
           query.exec();    + 1 (executes query)

           // get the highest id from the bottom row of the table
           if(query.next()) {     + 1 (Selection Statement) - if is bigger
              idNum =  query.value(0).toInt();     + 1 (gets the id number)
              idNum++;                             + 1 (increments id number)

              // Adds into campuses and distance into database
              query.prepare("INSERT INTO CAMPUSES VALUES(:ID, :START, :STOP, :DIST)");
                                                                  + 1 - prepares query
              query.bindValue(":ID", idNum); // id is the id of the bottom row + 1 + 1 (Adds ID)
              query.bindValue(":START", startDist);                + 1 (Add Campus)
```

```
            query.bindValue(":STOP", endDist);                          + 1 (Add Campus)
            query.bindValue(":DIST", distance);                 + 1 (Adds Distance)

            success = query.exec();     + 1 (Assigns variable)

            if(!success) {      + 1 (Selection Statement)
               qDebug() << "addCampus error: " << query.lastError();   + 1 (Displays Error)
            }

         } else {
            qDebug() << "Error: addCampus did not get an ID value from the table.";
         }
       }

      }
     // closes file
      inFile.close();     + 1 (Closes file)
  }
}
```

**Total Running Time for Add Campus(): O(n): (21n + 1) + 1 + 1 + 1 + 1 + 1 + 1 + 1 + 1 + 1 + 1 = 21n + 11**

**Function 3: getCampusNames() - O(n)**
Description: This function gets the campus names stored in the database and is pushed onto a vector.

Code:

```
vector<QString> DbManager::getCampusNames() {
  vector<QString> names;    + 1 (Declaring vector)

  // query database for campus names
  QSqlQuery query("SELECT DISTINCT START FROM CAMPUSES");   + 1 (Prepares query)

   // add campus names to vector (unique)
 Total Running Time: 2n + 1
  while(query.next()) {     +n + 1 (loop overhead)
     QString out = query.value(0).toString();   + 1 (Assigns variable)
     names.push_back(out);     + 1 (push function is always O(1) for STL vector)
  }

  return names;       + 1 (returns names vector)
}
```

**Total Running Time for getCampusNames(): O(n): (2n + 1) + 1 + 1 = 2n + 3**

**Function 4: PopulateTransactionTable() - O(n)**

Description: Populates the transaction table in order to display the Campus, Souvenir, Price and Quantity of each item purchased

Code:

```
void displaypurchases::populateTransactionTable(vector<Purchase>* purchaseList, const
QString& campus, bool displayAll)
{
  vector<Purchase> tempPurchases; // holds purchases that will be displayed on the table + 1
  double grandTotal = 0;          // the grand total for the currently selected campus(es)  + 1
  int rowCount = 0;               // the row that is currently having items added to it  + 1

  // leave function if no campus was provided
  if (!displayAll && campus == "") + 1(Selection)
  {
     return;
  }

  // set up the table of transactions
  ui->transactionTableWidget->setColumnCount(5);   +1 (Set Table)
  ui->transactionTableWidget->setColumnWidth(0, 275);  +1 (Set Table)
  ui->transactionTableWidget->setColumnWidth(1, 225); +1 (Set Table)
  ui->transactionTableWidget->setColumnWidth(2, 70); +1 (Set Table)
  ui->transactionTableWidget->setColumnWidth(3, 70); +1 (Set Table)
  ui->transactionTableWidget->setColumnWidth(4, 108); +1 (Set Table)
  ui->transactionTableWidget->setRowCount(1); +1 (Set Table)
  ui->transactionTableWidget->verticalHeader()->hide(); +1 (Set Table)
                                                        +1(Set Table)
  ui->transactionTableWidget->setHorizontalHeaderItem(0, new
QTableWidgetItem("Campus"));
  ui->transactionTableWidget->setHorizontalHeaderItem(1, new
QTableWidgetItem("Souvenir")); +1 (Set Table)
                                  +1(Set Table)
  ui->transactionTableWidget->setHorizontalHeaderItem(2, new QTableWidgetItem("Price"));
                                  +1(Set Table)
  ui->transactionTableWidget->setHorizontalHeaderItem(3, new
QTableWidgetItem("Quantity"));
                                  +1(Set Table)
  ui->transactionTableWidget->setHorizontalHeaderItem(4, new QTableWidgetItem("Total"));
```

```
ui->transactionTableWidget->setSortingEnabled(false); +1 (Set Table)

// save purchases that will be displayed on the table according to the campus name that was
// provided
Total Running Time(for loop): 3n + 1
for (auto it = (*purchaseList).begin(); it != (*purchaseList).end(); it++) +n + 1 (loop overhead)
{
    if (displayAll || it->campusName == campus) +1(Selection)
    {
        tempPurchases.push_back(*it);     +1(Push function for vector runs in O(1) time)
        grandTotal += it->totalSpent;      +1(Assigning variables)
    }
}

// display the grand total spent during the trip at the specified campus(es)
ui->totalLabel->setText("$" + QString::number(grandTotal));    +1 (Displays price)

// create and insert items into each individual cell of the transaction table
Total Running Time: 25(2n) = 50n + 1
for (auto it = tempPurchases.begin(); it != tempPurchases.end(); it++) n + 1 (loop overhead)
{
    rowCount++;    +1(increments row)
    ui->transactionTableWidget->insertRow(rowCount); +1(inserts row)

    // populating each column in row at rowCount with correct data
     Total Running Time: +5(3) + 5(2)  = 15 + 10 = 25
    for (int j = 0; j < 5; j++)   + 5
    {
        QTableWidgetItem *newItem = new QTableWidgetItem();  + 1 (allocating new widget)

        // set data for new item according to what column is currently being populated
        Total Running Time: +3
        switch (j)   +1(Selection statement)
        {
        case 0:
            newItem->setText( it->campusName );   + 1(Sets campus name)
            break; + 1(breaks out of case)
        case 1:
            newItem->setText( it->souvenirName );   + 1(Sets souvenir name)
            break; + 1(breaks out of case)
        case 2:
            newItem->setText("$" + QString::number(it->price) );   + 1(Sets price)
            break;   + 1(breaks out of case)
        case 3:
```

```
            newItem->setText( QString::number(it->numberPurchased) );   + 1(Sets quantity)
            break;   + 1(breaks out of case)
        case 4:
            newItem->setText("$" + QString::number(it->totalSpent) );  + 1(Sets total spent)
            break;   + 1(breaks out of case)
        default:
            break;
        }

        newItem->setTextAlignment(Qt::AlignHCenter);    + 1(setting text)

        // set the item on the table at row rowCount, column j
        ui->transactionTableWidget->setItem(rowCount, j, newItem);  + 1 (setting item)
    }
  }
}
```

**Total Running Time for PopulateTransactionTable: O(n): (3n+1) + (50n + 1) + 18 = 53n + 20**

**Function 5: SouvExists() - O(1)**
Description: This function checks if a souvenir is in the database.

Code:

```
bool DbManager::souvExists(QString &campus, QString &souvenirName)
{
  QSqlQuery query;  +1(Declaring variable)
  bool success;    +1(Declaring variable)
  bool found;    +1(Declaring variable)

  query.prepare("SELECT EXISTS(SELECT 1 FROM SOUVENIRS WHERE
CAMPUS=:CAMPUS AND SOUVENIR=:SOUVENIRNAME)");  +1(prepares query)
  query.bindValue(":CAMPUS", campus);          +1(Checks campus name)
  query.bindValue(":SOUVENIRNAME", souvenirName);    +1(Checks souvenir name)

  success = query.exec();    +1(Assigns variable)

  if(!success) {   +1(Selection Statement)
      qDebug() << "souvExists error: " << query.lastError();   +1(Displays Debug error)
      return false;   +1 (returns value)
  }

  query.first();    +1(gets the first value of a query)
  found = query.value(0).toBool();    +1(Assigns found to bool)
```

```
  return found;    +1(returns variable)
}
```

**Total Running Time for SouvExists: O(1) - 11**