Big ‘O’ Notation - Project 1

1.
```
Set MainWindow::getBasic()
{
   Set retVal;                                          //1
   for (unsigned long i =0;i< members.length();i++)    //n
   {
       if (members[i].isBasic())                        //n
           retVal.insert(members[i]);                   //n, insert() is O(1)
   }
   return retVal;                                                  //1
}
```
<u>O(n)</u>

2.
```
string MainWindow::getRebateReport()
{
   Set preferredMembers = getPreferred();                          //n, getPreferred() is O(n)
                                                                     same as getBasic()
   preferredMembers.sortById();                                    //n log(n), sortById() is
                                                                     O(nlog(n))
   string display;                                                 //1
   display += "Rebate Report:\n\n";                                //1
   for (size_t i =0;i<preferredMembers.length();i++)               //n
   {
       display += "(";                                             //n
       display += to_string(preferredMembers[i].getId());          //n, getId() is O(1)
       display +=") ";                                             //n
       display += preferredMembers[i].getName();                   //n, getName() is O(1)
       display +="\n";                                             //n
       display +="Rebate Amount: ";                                //n
       std::stringstream stream;                                   //n
       double rebate = preferredMembers[i].getTotalSpent()*.05;    //n, getTotalSpent() is O(1)
       stream << std::fixed << std::setprecision(2) << rebate;     //n
       display += stream.str();                                    //n
       display += "\n\n";                                          //n
   }
   if(preferredMembers.length() == 0) {                            //1
       display += "No preferred members\n";                        //1
   }
   return display;                                                 //1
}
```

Big ‘O’ Notation - Project 1

O(nlog(n)), n is the number of members

3.

```
string MainWindow::getAllMemberPurchases(char flag)
{
  if(flag != 'a' && flag != 'b' && flag != 'p') {                          //1
     return "Incorrect flag.\n";                                           //1
  }
  std::string display;                                                     //1
  display += "All Member Purchases\n\n";                                   //1
  Receipt totalReceipt;                                                    //1
  Set membersToQuery;                                                      //1
  if (flag == 'a')                                                         //1
     membersToQuery = members;                                             //n
  if (flag == 'b')                                                         //1
     membersToQuery = getBasic();                                          //n, getBasic() and =
                                                                             operator overload
                                                                             are both O(N)
  if (flag == 'p')                                                         //1
     membersToQuery = getPreferred();                                      //n, getPreferred()
                                                                             and = operator
                                                                             overload are both
                                                                             O(N)
  if(membersToQuery.length() == 0) {                                       //1
     return "No memebers\n";                                               //1
  }
  membersToQuery.sortById();                                               //nlog(n), sort is
                                                                             O(nlog(n))
  for(std::size_t i = 0; i < membersToQuery.length(); i++) {               //n
     Receipt memberTotalReceipt;                                           //n
     std::vector<Receipt> temp = membersToQuery[i].getReceipts();          //n, [] and getReceipts
                                                                             are both O(1)
     for(auto it = temp.begin(); it != temp.end(); it++) {                 //n*m, m is number of
                                                                             receipts in vector
        memberTotalReceipt += *it;                                         //n*m*p, p is number
                                                                             of items in receipt
        totalReceipt += *it;                                               //n*m*p
     }
     display += "(";                                                       //n
     display += to_string(membersToQuery[i].getId());                      //n
```

```
        display += ") ";                                                //n
        display += membersToQuery[i].getName() +="\n";                  //n
        if (temp.size() == 0)                                           //n
        {
            display += "No Purchases For This Member";                  //n
        }
        std::map<Item, int> items = memberTotalReceipt.getItems();      //n*p
        for(auto it = items.begin(); it != items.end(); it++) {         //n*p
            display += it->first.getName();                             //n*p
            display += " ($";                                           //n*p
            std::stringstream stream;                                   //n*p
            stream << std::fixed << std::setprecision(2) << it->first.getPrice();   //n*p
            std::string price = stream.str();                       //n*p
            display += price;                                           //n*p
            display += ")  x ";                                         //n*p
            display += to_string(it->second);                           //n*p
            display += '\n';                                            //n*p
        }
        display += "\n\n";                                              //n
    }
    display += "\n\n\nGrand Total: ";                                   //1
    std::map<Item, int> items = totalReceipt.getItems();                //p
    double grandTotal = 0;                                              //1
    for(auto it = items.begin(); it != items.end(); it++) {             //p
        grandTotal += (it->second)*(it->first.getPrice());              //p
    }
    std::stringstream stream;                                           //1
    stream << std::fixed << std::setprecision(2) << grandTotal;         //1
    std::string total = stream.str();                                   //1
    display += "$ ";                                                    //1
    display += total;                                                   //1
    return display;                                                     //1
}
```

O(n*m*p), n is the number of members, m is the number of receipts for a member, p is the number items in a receipt

= **O(n) * O(m) * O(p)**

4.

```
Set & Set::operator = (const Set& source)
{
  if (this == &source)                              //1
     return *this;                                  //1
  else                                              //1
  {
     capacity = source.capacity;                    //1
     size = source.size;                            //1
     delete [] data;                                //n
     data = new value_type[capacity];               //1
     for (size_type i = 0; i < size; i++)           //n
     {
        data[i] = source.data[i];                   //n
     }
  }
  return *this;                                     //1
}
```

//O(n)