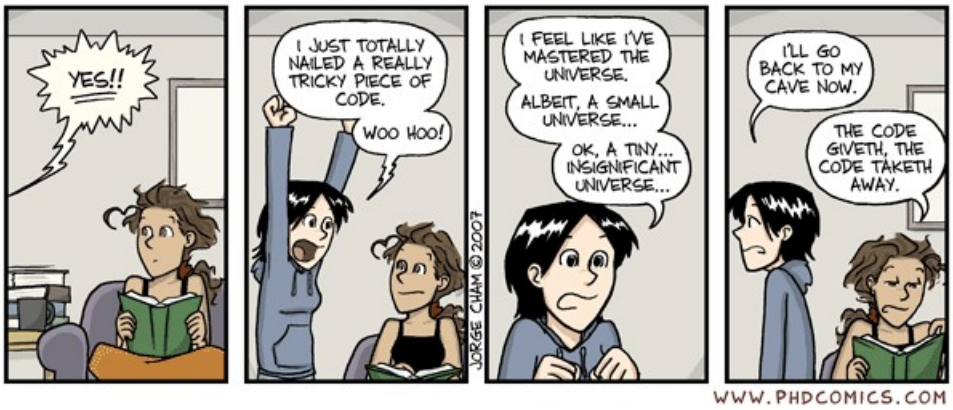
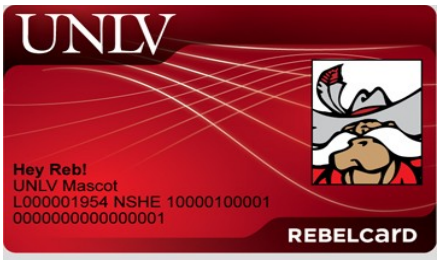
CS 202 – Assignment #3

Purpose: Learn class inheritance, composition, and multi-file class implementation.
Points: 100

Assignment:
Design and implement three C++ classes to provide a student, under graduate, and graduate classes. These classes will track information for students, both undergraduate and graduate.



The classes we will implement are as follows:



- **Student Class**
  - Implement a basic student class, ***student***, will track student information and provide standard support functions for all students. This will include name (first and last), ID, charges/fees, major, and grade point average (GPA). This will be the *base class* as this will apply to any student (undergraduate or graduate). The other classes will inherit this functionality, so this class must be fully working before working on the subsequent classes.

- **Money Class**
  - A small class, ***money***, the will developed that contains only the student financial data and functions. This isolates the financial information and allows for special protections (encryption) and reporting as is typically required for state and federal reporting. Such functionality will not be part of this project.

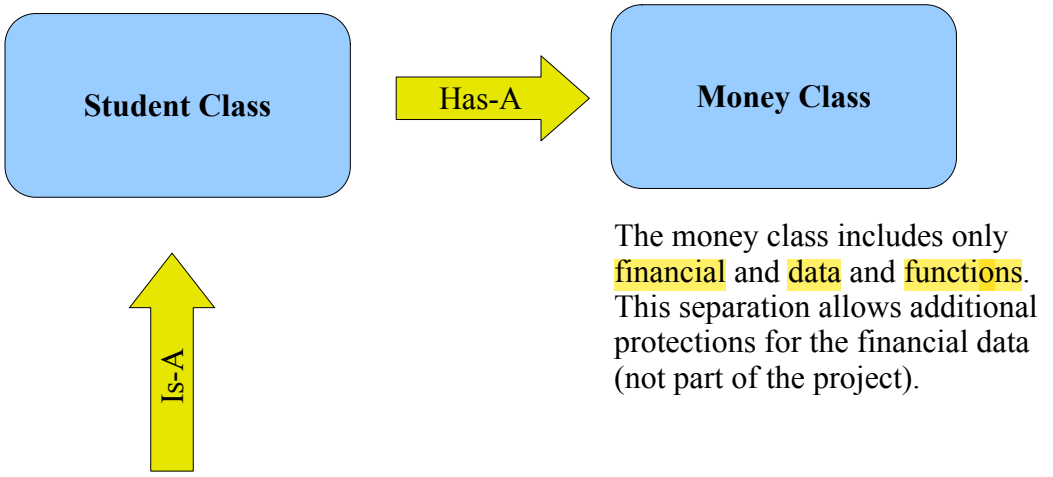
- **Undergraduate Student Class**
  - The undergraduate class, ***underGrad***, will provide the same basic capabilities as the *student* class with some additional functionality. The additional functionality will include tracking the assigned adviser and the student states (e.g., good, special, probation). By using inheritance and deriving this class from the student class, this will ensure we do not have the re-implement the code from the student class.

- **Graduate Class**
  - The graduate class, ***grad***, will provide the same basic capabilities as the ***underGrad*** class with some additional functionality. The additional functionality will include tracking the graduate fees and the graduate assistantship status. Again, by using inheritance and deriving this class from the undergraduate class, this will ensure we do not have the re-implement the code for either of the previous classes.

For all implementation code, points will be deducted for poor or especially inefficient solutions.

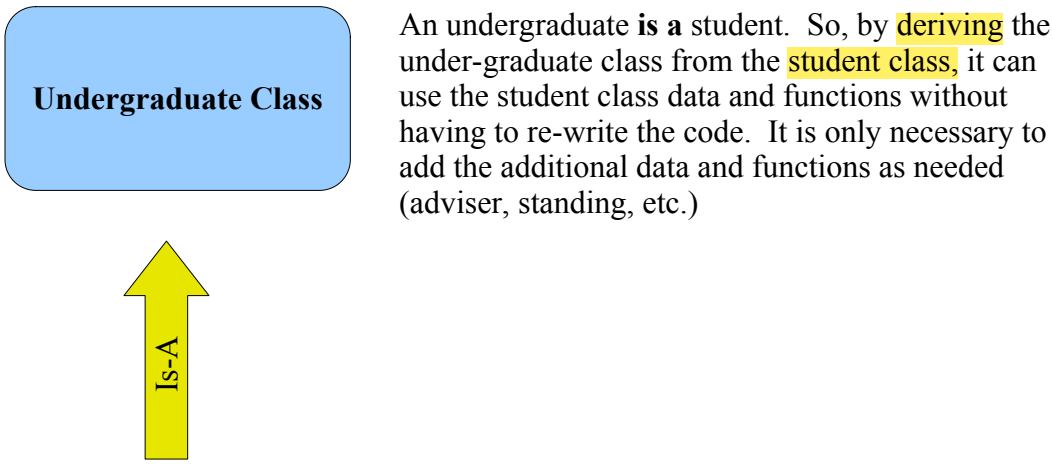
Class Hierarchy

The following diagram should help understand the class hierarchy for this project.

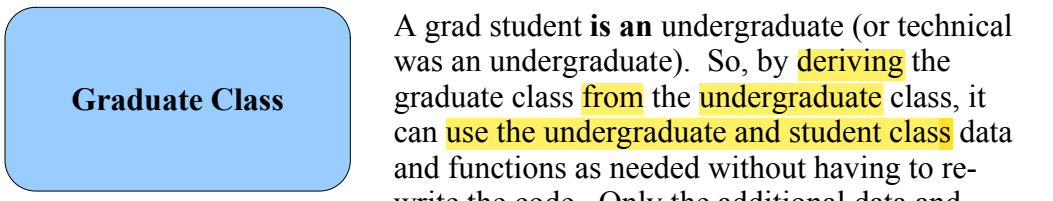


The student base class encompasses basic student data and functions.

The money class includes only financial and data and functions. This separation allows additional protections for the financial data (not part of the project).



An undergraduate **is a** student. So, by deriving the under-graduate class from the student class, it can use the student class data and functions without having to re-write the code. It is only necessary to add the additional data and functions as needed (adviser, standing, etc.)



A grad student **is an** undergraduate (or technical was an undergraduate). So, by deriving the graduate class from the undergraduate class, it can use the undergraduate and student class data and functions as needed without having to re-write the code. Only the additional data and functions need be added.

Inheritance and composition and fundamental concepts in object oriented programming and it its important to fully understand. Refer to the text and class lectures for additional information regarding inheritance and composition.

Development and Testing

In order to simplify development and testing, the project is split into three (3) main parts; student and money classes, undergraduate class, and graduate class.

- The student and money classes must be developed first (before the derived classes). These classes can be developed and tested independently of the other classes. An independent main that uses and tests the student and money classes is provided and can be built independently using the provided main file (see next section). *Note*, this is the largest portion of the project.
- The undergraduate class will use the student and money classes so they should be fully working and tested before starting the undergraduate class. However, the undergraduate class development and testing can be developed and tested independently of the graduate class. An independent main that uses and tests the undergraduate class (which use the student and money classes) is provided and can be built independently using the provided main file (see next section).
- The graduate class will use the student, money, and undergraduate classes so they should be fully working and tested before starting the undergraduate class. A provided main that uses and tests the graduate class (which use the student, money, and undergraduate classes) is provided and can be built using the provided main file (see next section).

Make File:

The provided make file assumes the source files are named in accordance with the class names. See class descriptions (at end) for expected file names. To build the student class, undergraduate class, or graduate class provided testing mains;

```
make stdMain
make underMain
make gradMain
```

Which will create the *stdMain* (for *student* class), *underMain* (for *underGrad* class) and the *gradMain* (for the *grad* class) executables respectively.

Submission:

- It is **strongly** suggested that you submit each part of the program for testing as they are developed. For example, when the student and money classes are completed, they can be submitted which will allow testing of that portion of the project. This will ensure it is fully working before moving on to the next part. If you find any issues, they can be corrected and resubmitted an unlimited number of times (before the due date/time).
- All files must compile and execute on Ubuntu and compile with C++11.
- Submit source files
  - *Note*, do ***not*** submit the provided mains (we have them).
- Once you submit, the system will score the project and provide feedback.
  - If you do not get full score, you can (and should) correct and resubmit.
  - You can re-submit an unlimited number of times before the due date/time.
- Late submissions will be accepted for a period of 24 hours after the due date/time for any given lab. Late submissions will be subject to a ~2% reduction in points per an hour late. If you submit 1 minute - 1 hour late -2%, 1-2 hours late -4%, … , 23-24 hours late -50%. This means after 24 hours late submissions will receive an automatic 0.

Program Header Block

All program and header files must include your name, section number, assignment, NSHE number, input and output summary. The required format is as follows:

```
/*
Name: MY_NAME, NSHE, CLASS-SECTION, ASSIGNMENT
Description: <per assignment>
Input: <per assignment>
Output: <per assignment>
*/
```

Failure to include your name in this format will result in a loss of up to 5%.

Code Quality Checks

A C++ linter[1] is used to perform some basic checks on code quality. These checks include, but are not limited to, the following:

- Unnecessary 'else' statements should not be used.
- Identifier naming style should be either camelCase or snake_case (consistently chosen).
- Named constants should be used in place of literals.
- Correct indentation should be used.
- Redundant return/continue statements should not be used.
- Selection conditions should be in the simplest form possible.
- Function prototypes should be free of top level *const*.

Not all of these items will apply to every program. Failure to to address these guidelines will result in a loss of up to 5%.

Scoring Rubric

Scoring will include functionality, code quality, and documentation. Below is a summary of the scoring rubric for this assignment.

Criteria	Weight	Summary
Compilation	-	Failure to compile will result in a score of 0.
Program Header	5%	Must include header block in the required format (see above).
General Comments	10%	Must include an appropriate level of program documentation.
Line Length	5%	No lines should exceed more than eighty (80) characters.
Code Quality	5%	Must meet some basic code quality checks (see above)
Program Functionality		Program must meet the functional requirements as outlined in the assignment. Must be submitted on time for full score.
• Student/Money Classes	30%	
• Undergraduate Class	25%	
• Graduate Class	20%	

1 For more information, refer to: https://en.wikipedia.org/wiki/Lint_(software)

Student Class

Implement a basic student class named ***student*** to provide student functions. The *student* UML class diagram is as follows:

student
-lastName, firstName: string
-studentID, major: string
-gpa: double
-majorsFile: ifstream
-majorCodesCount: int
-majorCodes[MAX_MAJORS]: string
-finances: money
-MAX_MAJORS = 250: static constexpr int
-MAX_GPA = 4.0: static constexpr double
+student(string="", string="", string="", string="", double=0.0, double=0.0, double=0.0)
+~student()
+getLastName() const: string
+getFirstName() const: string
+getID() const: string
+getMajor() const: string
+getGPA() const: double
+getCharges(double &, double &): void
+getBalance() const: double
+setName(string, string): void
+setID(string): void
+setMajor(string): void
+setGPA(double): void
+setCharges(double, double): void
+showStudent(): void
-checkID(string) const: bool
-expandMajorCode(string): string

The allowed majors for the *expandMajorCode(string)* function are provided in a file. This file must be read and an array populated with this information.

Function Descriptions:

The following are more detailed descriptions of the required functions.

- Function *student()* is the constructor. Initializes the class variables to the passed arguments (if any) or the default values. The arguments, in order, are last name (string), first name (string), student ID (string), student major code (string), grade point average (double), charges (double), and financial aid (double). The constructor must call the money constructor function with the charges and financial aid values. The constructor must check the passed values for errors as follows:

- The student ID string must be a UNLV formatted L-number. The *checkID(string)* function will check the format (returning a boolean). If not valid, an error message should be displayed and the student ID variable set the empty string.
- The student major code is a 2, 3, or 4 character string and must be one of the legal/allowed values. The *expandMajorCode(string)* function will perform this check and expansion (from code to the full name).
- The grade point average must be between 0.0 and MAX_GPA (inclusive). If not valid, an error message should be displayed and the GPA variable set to 0.0.

Additionally, the constructor should open the majors file and read the majors list into the majorCodes[] array and set the majorCodesCount variable accordingly. The majors file name can be hard coded as "majorsList.txt".

- Function *~student()* is the destructor. For this project, the destructor only needs to close the file.
- Function *getLastName()*, function *getFirstName()*, and function *getID()* return the last name, first name, or student ID strings respectively.
- The *getMajor()* and *getGPA()* functions return the student student major and student grade point average respectively.
- The *setName(string, string)* function sets the last name and first name (in that order).
- The *setID(string)*, *setMajor(string)*, and *setGPA(double)* functions set the student ID, student major, and student grade point average based on the passed values. This includes error checking. If invalid values are provided, an error message should be displayed and the applicable class variable not altered. The error checks are described above.
- The *showStudent()* function should display a student summary in the format shown below. The values should line up (as shown). The monetary values must be displayed as shown with two decimal points (as is standard).

```
--------------------------------------------------------------------
Student Summary
        Name: Leonard, Lenny
          ID: L000018716
       Major: NUE - Nuclear Engineering
         GPA:     3.20
     Charges:   975.25               Financial Aid:     0.00
     Balance:   975.25
```

The following are the **cout** statements were used to create the formatted output.

```
string indent(5, ' ');
string bars;
bars.append(70, '-');

cout << endl << bars << endl;
cout << "Student Summary" << endl;
cout << indent << " Name: " << lastName << ", " <<
                  firstName << endl;
cout << indent << " ID: " << studentID << endl;
cout << indent << " Major: " << major << endl;
cout << fixed << showpoint << setprecision(2);
cout << indent << " GPA: " << setw(8) << gpa << endl;
cout << indent << "Charges: " << setw(8) << ch << indent
            << indent << indent << "Financial Aid: " <<
            setw(8) << fn << endl;
cout << indent << "Balance: " << setw(8) <<
finances.getBalance() << endl;
```

- The *checkID(string)* function should check a student ID string. A valid ID must start with an 'L' followed by exactly 9 digits ('0'-'9'). The function should return true if valid, and false if not valid.

- The *expandMajorCode(string)* utility function should check the passed major code (2-4 characters) and check it against the list of valid major codes and return the major string which includes the major code and major name. The list of valid major codes/names will be provide. If the passed major code in not valid, the function should return the string "Error".

Money Class

Implement a basic student financial class named ***money*** to provide student financial support functions. The *money* UML class diagram is as follows:

money
-charges, financialAid, balance: double
-MAX_CHARGES = 10000.0: static constexpr double
+money(double=0.0, double=0.0)
+getCharges(double &, double &) const: void
+getBalance() const: double
+setCharges(double, double): void

The allowed majors for the *expandMajorCode(string)* function are provided in a file. This file must be read and the array populated with this information.

Function Descriptions:

The following are more detailed descriptions of the required functions.

- Function *money()* is the constructor. Initializes the class variables to the passed arguments (if any) or the default values. The arguments, in order, are charges (double) and financial aid (double). The constructor must check for errors as follows:
  - The charges must be between 0.0 and MAX_CHARGES (inclusive). If not valid, an error message should be displayed and the charges variable set to 0.0.
  - The financial aid must be between 0.0 and the provided value for student charges (inclusive). If not valid, an error message should be displayed and the financial aid variable set to 0.0.
  - The balance class variable should be set to the difference between the charges and financial aid.
- Function *getCharges(double &, double &)* returns the student charges and financial aid (via reference).
- The *getBalance()* function returns the student balance.
- The *setCharges(double, double)* function should set the student charges and financial aid class variables. This includes error checking. If invalid values are provided, an error message should be displayed and the applicable class variable not altered. The error checks are described in the constructor.

Use the provided main, *stdMain.cpp*, to demonstrate that the *student* class functions correctly. The main and provided makefile reference files *money.h*, *moneyImp.cpp*, *student.h* and *studentImp.cpp*. Your implementation and header files should be fully commented.

Undergraduate Class

The undergraduate class, ***underGrad***, will provide the same basic functionality and the *student* class with some additional functionality. Specifically, the *underGrad* class will add some additional capabilities. Since the *underGrad* will extend the *student* class functionality, *underGrad* will be derived form the *student* class.

The underGrad class header file should include the following enumeration:

```
enum sStat{PROBATION, GOOD, SPECIAL, NONE};
```

The ***underGrad*** UML class diagram is as follows:

underGrad
-advisor: string
-sStatus: sStat
+underGrad (string="", string="", string="", string="", double=0.0, double=0.0, double=0.0, string="", sStat=NONE)
+getAdvisor() const: string
+getStatus() const: sStat
+setAdvisor(string): void
+setStatus(sStat): void
+showStudent(): void

Function Descriptions:

The following are more detailed descriptions of the required functions.

- Function *underGrad()* is the constructor. Initializes the class variables to the passed arguments (if any) or the default values. The arguments, in order, are last name (string), first name (string), student ID (string), student major code (string), grade point average (double), charges (double), financial aid (double), student advisor (string), and student status (sStat, enumeration as defined above). The constructor must use the base class constructor.
- Function *getAdvisor()* function *getStatus()* functions return the advisor string and student status enumeration respectively.
- The function *setAdvisor(string)* function sets the advisor to the passed parameter and the and *setStatus(sStat)* function sets the student status to the passed parameter.
- The function *showStudent()* should display a student summary in the format shown below using the base class function. After that, some additional information should be displayed; adviser and status (as show below). If the student status is NONE, the print function should update the status; if the grade point average is ≤ 1.7, the status should be set to PROBATION, otherwise it should be set to GOOD.

```
---------------------------------------------------------------------
Student Summary
        Name: Quimby, Joe
          ID: L000127318
       Major: POS - Political Science
         GPA: 1.77
     Charges: 7120.50                Financial Aid: 800.00
     Balance: 6320.50
     Advisor: Gewali                 Status: PROBATION
```

Below are some **cout** statements that were used to create the last line.

```
string indent(5, ' ');

cout << indent << "Advisor: " << left << setw(17) << advisor;
cout << indent << " Status: ";
```

Use the provided main, *underMain.cpp*, to demonstrate that the ***underGrad*** class functions correctly. The provided main and provided makefile reference files *underGradImp.cpp* and *underGrad.h*. Additionally, this uses the *money.h*, *moneyImp.cpp*, *student.h* and *studentImp.cpp* files.
The student class must be fully completed prior to moving on to the *underGrad* class. Refer to the example executions for formatting. Make sure your program includes the appropriate documentation.

Graduate Class

The graduate class, ***grad***, will provide the same basic functionality and the *underGrad* class with some additional functionality. Specifically, the *grad* class will add some additional capabilities. Since the *grad* will extend the *underGrad* class functionality, *grad* will be derived form the *underGrad* class. The ***grad*** UML class diagram is as follows:

grad
-isGA: bool
-gradFees: double
-MAX_FEES = 5000.0: static constexpr double
+grad (string="", string="", string="", string="", double=0.0, double=0.0, double=0.0, string="", sStat=NONE, bool=false, double=0.0);
+getGAstatus() const: bool
+getGradFees() const: double
+setGAstatus(bool): void
+setGradFees(double): void
+showStudent(): void

Function Descriptions:

The following are more detailed descriptions of the required functions.

- Function *grad()* is the constructor. Initializes the class variables to the passed arguments (if any) or the default values. The arguments, in order, are last name (string), first name (string), student ID (string), student major code (string), grade point average (double), charges (double), financial aid (double), student adviser (string), student status (sStat, enumeration as defined above), the graduate assistance flag (bool), and the graduate fees (double). The constructor must use the base class constructor. The constructor must check for errors as follows:
  - The graduate fees must be between 0.0 and MAX_FEES (inclusive). If not valid, an error message should be displayed and the graduate fees variable set to 0.0.
- Function *getGAstatus()* function *getGradFees()* functions return the graduate assistance flag (bool) and graduate fees amount (double) respectively.
- The function *setGAstatus(bool)* function sets the graduate assistance flag (bool) to the passed parameter.
- The *setGradFees(double)* function sets the graduate student fees to the passed parameter. This includes error checking. If invalid values are provided, an error message should be displayed and the applicable class variable not altered. The error checks are described in the constructor.

- The function *showStudent()* should display a student summary in the format shown below using the base class function. After that, some additional information should be displayed; grad fees GA status. An example is as follows:

```
-------------------------------------------------------------------
Student Summary
        Name: Riviera, Nick
          ID: L000000666
       Major: NUC - Nuclear Medicine
         GPA: 1.00
     Charges: 9200.25                Financial Aid: 1500.00
     Balance: 7700.25
     Advisor: Jorgensen              Status: GOOD
   Grad Fees: 2100.20                   Graduate Assistant: Yes
```

The cout statements used to add the grade fees are

```
string indent(5, ' ');

cout << " Grad Fees: " << fixed << setprecision(2) <<
                  gradFees << indent;
cout << indent << indent << " Graduate Assistant: ";
```

Use the provided main, *gradMain.cpp*, to demonstrate that the *grad* class functions correctly. The provided main and provided makefile reference files *gradImp.cpp* and *grad.h*. Additionally, this uses the *money.h*, *moneyImp.cpp*, *student.h*, *studentImp.cpp* , undergrad.*h*, and *undergradImp.cpp* files. Your implementation and header files should be fully commented.

Example Execution:

Below is an example program execution output for the *gradMain* program.

```
ed-vm% ./gradMain

-------------------------------------------------------------------
CS 202 - Assignment #6
Graduate Class -> Test Program.

Error, invalid student ID.
Error, invalid student major.
Error, invalid student major.
Error, invalid student ID.
Error, invalid student major.
Error, invalid student ID.
Error, invalid student major.
Error, invalid student ID.
Error, invalid student major.
Error, invalid student major.
Error, invalid student major.
Error, invalid student ID.
Error, invalid student major.
Error, invalid student ID.
Error, invalid student ID.
Error, invalid student major.
Error, invalid student major.
Error, invalid GPA.
Error, invalid GPA.
Error, invalid charges amount.
Error, invalid financial aid amount.
Error, invalid student ID.
```

```
Error, invalid student major.
Error, invalid student ID.
Error, invalid student ID.
Error, invalid student major.
Error, invalid student major.
Error, invalid GPA.
Error, invalid GPA.
Error, invalid charges amount.
Error, invalid financial aid amount.
Error, invalid student ID.
Error, invalid student major.
Error, invalid student ID.
Error, invalid student ID.
Error, invalid student major.
Error, invalid student major.
Error, invalid GPA.
Error, invalid GPA.
Error, invalid charges amount.
Error, invalid financial aid amount.
Error, invalid student ID.
Error, invalid student major.
Error, invalid student ID.
Error, invalid student ID.
Error, invalid student major.
Error, invalid student major.
Error, invalid GPA.
Error, invalid GPA.
Error, invalid charges amount.
Error, invalid financial aid amount.
Error, invalid graduate fees.
**********************************************************************
Student Summary

----------------------------------------------------------------------
Student Summary
        Name: Simpson, Homer
          ID: L000001254
       Major: NUE - Nuclear Engineering
         GPA:     1.00
     Charges:     0.00               Financial Aid:     0.00
     Balance:     0.00

----------------------------------------------------------------------
Student Summary
        Name: Leonard, Lenny
          ID: L000018716
       Major: NUE - Nuclear Engineering
         GPA:     3.20
     Charges:   975.25               Financial Aid:     0.00
     Balance:   975.25

----------------------------------------------------------------------
Student Summary
        Name: Carlson, Carl
          ID: L000022716
       Major: NUE - Nuclear Engineering
         GPA:     2.75
     Charges:  2221.50               Financial Aid:     0.00
     Balance:  2221.50

----------------------------------------------------------------------
Student Summary
        Name: McClure, Selma
```

```
          ID: L000011116
       Major: UNS - University Studies
         GPA:     2.25
     Charges:  1400.75               Financial Aid:     0.00
     Balance:  1400.75

---------------------------------------------------------------------
Student Summary
        Name: Nahasapeemapetilon, Apu
          ID: L000012123
       Major: FOL - Foreign Language
         GPA:     3.45
     Charges:  1600.00               Financial Aid:     0.00
     Balance:  1600.00

---------------------------------------------------------------------
Student Summary
        Name: Groundskeeper, Willie
          ID: L000087290
       Major: PHY - Physics
         GPA:     2.50
     Charges:  2900.00               Financial Aid:  1000.00
     Balance:  1900.00

*********************************************************************
Undergraduate Summary

---------------------------------------------------------------------
Student Summary
        Name: McClure, Selma
          ID: L000011116
       Major: Error
         GPA:     2.25
     Charges:  1400.75               Financial Aid:     0.00
     Balance:  1400.75
     Advisor: Lee                    Status: GOOD

---------------------------------------------------------------------
Student Summary
        Name: Meyers, Roger
          ID: L000134341
       Major: Error
         GPA: 3.41
     Charges: 1300.75                Financial Aid: 350.00
     Balance: 950.75
     Advisor: Bien                   Status: GOOD

---------------------------------------------------------------------
Student Summary
        Name: Monroe, Marvin
          ID: L000120001
       Major: ART - Art
         GPA: 1.00
     Charges: 3700.75                Financial Aid: 1000.00
     Balance: 2700.75
     Advisor: Lee                    Status: PROBATION

---------------------------------------------------------------------
Student Summary
        Name: Wiggum, Clancy
          ID: L000000007
       Major: CRJ - Criminal Justice
         GPA: 0.23
```

```
     Charges: 2300.50                Financial Aid: 0.00
     Balance: 2300.50
     Advisor: Lee                    Status: PROBATION

------------------------------------------------------------------------
Student Summary
        Name: Ziff, Artie
          ID: L001201101
       Major: CS - Computer Science
         GPA: 4.00
     Charges: 9500.50                Financial Aid: 2000.00
     Balance: 7500.50
     Advisor: Yfantis                Status: GOOD

************************************************************************
Graduate Summary

------------------------------------------------------------------------
Student Summary
        Name: McClure, Selma
          ID: L000011116
       Major: Error
         GPA: 2.25
     Charges: 1400.75                Financial Aid: 0.00
     Balance: 1400.75
     Advisor: Lee                    Status: GOOD
   Grad Fees: 235.75                   Graduate Assistant: No

------------------------------------------------------------------------
Student Summary
        Name: Burns, Charles
          ID: L000017254
       Major: UND - Undeclared
         GPA: 2.50
     Charges: 0.00                   Financial Aid: 0.00
     Balance: 0.00
     Advisor: Jorgensen              Status: PROBATION
   Grad Fees: 1000.00                   Graduate Assistant: Yes

------------------------------------------------------------------------
Student Summary
        Name: Quimby, Joe
          ID: L000127318
       Major: POS - Political Science
         GPA: 1.77
     Charges: 7120.50                Financial Aid: 800.00
     Balance: 6320.50
     Advisor: Gewali                 Status: PROBATION
   Grad Fees: 1234.56                   Graduate Assistant: Yes

------------------------------------------------------------------------
Student Summary
        Name: Hibbert, Julius
          ID: L000087346
       Major: PBH - Public Health
         GPA: 3.90
     Charges: 1112.50                Financial Aid: 1000.00
     Balance: 112.50
     Advisor: Vasko                  Status: GOOD
   Grad Fees: 1200.02                   Graduate Assistant: Yes

------------------------------------------------------------------------
Student Summary
```

```
        Name: Chalmers, Gary
          ID: L000001254
       Major: HOA - Hotel Administration
         GPA: 3.10
     Charges: 1000.00                Financial Aid: 500.00
     Balance: 500.00
     Advisor: Taghva                 Status: GOOD
   Grad Fees: 2344.67                   Graduate Assistant: Yes

-------------------------------------------------------------------------
Student Summary
        Name: Krabappel, Edna
          ID: L000021231
       Major: WOM - Women's Studies
         GPA: 3.15
     Charges: 1500.00                Financial Aid: 1000.00
     Balance: 500.00
     Advisor: Kim                    Status: GOOD
   Grad Fees: 200.12                   Graduate Assistant: No

-------------------------------------------------------------------------
Student Summary
        Name: Riviera, Nick
          ID: L000000666
       Major: NUC - Nuclear Medicine
         GPA: 1.00
     Charges: 9200.25                Financial Aid: 1500.00
     Balance: 7700.25
     Advisor: Jorgensen              Status: GOOD
   Grad Fees: 2100.20                   Graduate Assistant: Yes

*************************************************************************
ed-vm%
```