CIS1500 Assignment 2: Student Finance Calculator

Total Marks: 10 marks
Weight: 10%
Due: Friday, 3rd Nov 2023, at 11:59 pm

Welcome to your second CIS1500 assignment! Please ensure you read all the sections and **follow the instructions exactly** as we will grade your submission with an auto-grader. You must **test your assignment on the SoCS Linux server**, and it will be similarly graded on the SoCS Linux server for consistency.

Your program should only use concepts already taught in this course. **You will receive a 0 if your code includes: global variables** (variables declared outside of functions), **goto, malloc, calloc, regex, or if your code fails to compile without errors.** Note there are already two global **constants** provided for you in the .h file.

This pdf goes in conjunction with videos available on courselink.

There is also a file containing "Skeleton Code"; you should use this as your starting point. It contains 3 files:

1) A .c file for your main called **skeleton.c. You must rename this before submitting.**
2) A .c file for your functions called **A2_Functions.c. Do not rename this before submitting.**
3) A .h file for function prototype called **A2_Functions.h. This file is already completed, do not modify it.** You do not need to submit it.

This document has 15 pages. Read them all and understand the requirements.

REMEMBER: You are to submit **one zip file containing two source code files** for your program.

- The name of the **zip file** follows the following format: **studentNumberA2.zip**
  - Example: John Snow's student number is 1770770 then the zip file name is 1770770A2.zip
- The two files within the zip you are to submit are non-compiled code C files (ends with .c) containing your code
  - Do not include any other files or folders in your submission; they will be ignored.
  - Do not include this file inside another folder; it will be ignored by the auto-grader.
- The name of the **C file** containing your main follows the following format: **studentNumberA2.c**
  - Example: John Snow's student number is 1770770 then the C file name is 1770770A2.c
- The name of the C file containing your functions is A2_Functions.c
- **Incorrect submission will result in a grade of 0.**
- More details are outlined in Section 4: Program Submission and Administration Information.

Start Early. Get Help Early. Test Often.

1. Background

Please refer to the A1 outline for background, as this assignment is heavily based on A1. You need to thoroughly understand both assignment outlines. In this assignment we will make 3 major changes:

- If a user enters an invalid input the **message "Invalid Input"** is displayed, then the user is asked to enter the value again. This requires loops.
- We have 9 mandatory functions, there are functions to get an input, get multiple inputs, print the finance table, and print the finance summary.
- The "Food" and "Other Expenses" prompts are removed, instead you will have arrays storing expense names and expense values, as well as a counter of how many expense items were added.

The output of your program should look very similar to that of A1 with three exceptions:

- The prompt for expenses has changed, and can repeat up to `MAX_NUM_EXPENSES`
- We have added a '\n' between each input prompt, this helps with readability, and with grading. (The skeleton code already has an example function containing a printf with the newline in the correct place).
- The finances table may have more or less lines depending on the number of expenses input by the user.

This program is intended to test your ability to:

- Use conditional and counting loops.
- Create, modify, and print arrays.
- Create functions
    - pass by value
    - pass by reference
    - pass arrays

The course and assignment are not intended as actual finance advice.

2. Program Requirements

Your program will start with a welcome message, then prompt the user to answer some questions about savings and income, then ask questions about expenses; after getting the user information the program will display a table with calculated finance values, and finally display a few sentences summarizing the information.

2.1 Required Functions

This assignment has 9 mandatory functions:

1) **int getIntInput(char prompt[]);**
    - This functions has one argument, a string called prompt: this is a message to be displayed to the user. The message provides instructions on what to input.
    - The printf statement that displays the prompt should start with \n.
        - printf("\n%s", prompt);
        - This can be seen in the start
    - The function returns an integer - the value entered by the user.

- If the user enters a negative value, the **message "Invalid Input.\n"** is displayed, then print the prompt again. It will continue to ask for input until the user enters a non-negative value.

2) **float getFloatInput(char prompt[]);**
   - Same as getIntInput except the function reads and returns a float instead of an int.

3) **char getCharInput(char prompt[], char valid_inputs[], int num_valid);**
   - This function has 3 arguments:
     - a string called prompt: see getIntInput details.
     - an array of characters called valid_inputs. These are the acceptable characters a user may enter. Ex. {'Y','N'}
     - an integer num_valid. This is the size of the valid_inputs array. (note valid_inputs is not a string, so it does not need space for a null character).
   - The function returns a char - the value entered by the user.
   - If the user enters a character that is not in the valid_inputs array, the **message "Invalid Input.\n"** is displayed, then print the prompt again. It will continue to ask for input until the user enters a character in valid_inputs
   - The function should be case-insensitive, meaning that if an uppercase character is in the valid_inputs array, then either the lowercase or uppercase version should be accepted as well. Also if a lowercase character is in the array, both cases are accepted. Hint: use tolower or toupper from ctype.h

4) **float getWeeklyPay();**
   - This function will use the above get function(s) to prompt the user for inputs and retrieve information.
   - The function will ask if the user has a part time job, if yes it will ask for hourly wage and number of hours worked in a week. See the A1 outline and the sample flow images for the exact prompt text.
   - The prompt text, and order of questions must match exactly.
   - The function returns a float. If the user states they have no part time job then a value of 0.0 is returned. Otherwise it will return the hourly wage multiplied by the number of hours worked per week.
5) **float getAnnualAllowance();**
   - This function will use the above get function(s) to prompt the user for inputs and retrieve information.
   - The function will ask if the user receives an allowance, if yes it will ask for the amount and frequency. See the A1 outline and the sample flow images for the exact prompt text.
   - The function returns a float. If the user states they do not receive an allowance then a value of 0.0 is returned. Otherwise it will return the total annual amount of allowance given the amount payment amounts and frequency.

6) **void getHousing(float *rent, float *utilities);**
    - This function has 2 arguments, both pass-by-reference:
        - a float called rent.
        - a float called utilities
        - These will be set based on user input, they will be set to 0 if the user states that they do not pay for housing.
    - This function will use the above get function(s) to prompt the user for inputs and retrieve information.
    - The function will ask if the user pays for housing, if yes it will ask for the amount and frequency. See the A1 outline and the sample flow images for the exact prompt text.
    - The function returns a float. If the user states they do not pay housing then a value of 0.0 is returned. Otherwise it will return the yearly amount.
    - Hint** look at chapter 5b, and w7 lecture slides to learn more about pass-by-reference.

7) **void getExpenses(char expense_names[MAX_NUM_EXPENSES][MAX_STRING_LEN],**
   float expense_costs[MAX_NUM_EXPENSES],
   int *num_expenses,
   float *expenses_total);
    - This function will retrieve expenses from the user and store them in arrays, and variables passed by reference. The expenses arrays replace the “food” and “other expenses” categories from A1.
    - This function has 4 arguments:
        - an array of strings called expense_names: this will store the names of expenses entered by the user
        - an array of floats called expense_costs: this will store the cost of the expenses entered by the user
        - an int called num_expenses pass-by-reference: this will store the length of the two arrays after being populated with user data.
        - a float called expenses_total pass-by-reference: this will store the sum of the expenses in the expense_costs array.
        - These will be set based on user input, they will be set to 0 if the user states that they do not pay for housing.
    - Your program will prompt ‘Enter a one-word name of an expense, if you are finished enter \"finished\": ’
    - If the user did not enter “finished” will ask ‘What is the monthly cost of this expense? $ ’
    - The user must enter ‘finished’ not ‘Finished’, ‘FINISHED’ or other variation.
    - This is repeated until either the user enters “finished” or num_expenses reaches MAX_NUM_EXPENSES
    - If num_expenses reaches MAX_NUM_EXPENSES print “\nMaximim Expenses Reached\n”, and move on to printing Finance Table
    - The one-word name of the expense is stored in the expense_names array, and the cost is stored in the expense_costs array.
    - Make certain to update the num_expenses and expenses_total variables.
    - Hint** to compare the input against “finished” use the strcmp function from string.h. This function returns 0 if two strings are identical.

8) **void printTable(float savings,**
float grants,
float net_funds,
float monthly_funds,
float monthly_income,
float monthly_budget,
float rent,
float utilities,
char expense_names[MAX_NUM_EXPENSES][MAX_STRING_LEN],
float expense_costs[MAX_NUM_EXPENSES],
int num_expenses,
float essentials_total,
float monthly_essentials_avail,
float expensesRemaining,
float monthly_wants,
float monthly_savings
);
   - This function has a large number of inputs. The names should indicate their purpose.
   - This function will not have any calculations, the values are all printed with the value passed to the function.
   - The printing format is the same as A1. With the exception that "Food" and "Other Expenses" is replaced by the values in expense_names and expense_costs.
   - Note the skeleton code has formatting example for non-indented rows with costs, this was missing in the first skeleton code.

9) **void printSummary(float expensesRemaining, float wantsAvailable, float savingsAvailable);**
   - The final function required, prints the summary just as shown in A1.
   - Just as in A1, if expensesRemaining is negative print "You need to earn an additional …" otherwise "You are following the 50/30/20 rule …"
   - If the wantsAvailable or savingsAvailable values are below 0 set them to 0 instead.

Tips:
* Always start small. All these functions sound daunting at first, pick one and start with it; getFloatInput would be a good option. You can call this function from the main, store the returned value in a variable, and print it. This will allow you to test just the individual function.
* Thoroughly test a function before moving on to the next, as many functions use others, it may be hard to trace the source of the error if one of the more basic functions has an error.
* Remember Edit-Compile-Run: regularly test your code by compiling and running it.

* Once you figured out how to do one getFloatInput, think about how to copy and modify it to create getIntInput. Repeat the steps from the previous tips.

* Converting your table and summary code to a function may also be quite simple, you likely need only to rename a few variables and copy your existing printf code to the appropriate function.

2.2 Part 1: Getting User Input

You must use the appropriate functions to retrieve input from the user, the main should not contain any scanf calls, and the only printf should be the initial "Welcome to the CIS1500 Student Finance Planner…" message.

Your program will prompt the user for the same questions as A1, except the getExpenses function will be used to read custom expenses from the user in place of "Food" and "Other Expenses". Remember that the expense names must be a single word (no spaces), and the user must enter "finished" to stop entering expenses (no capitals).

Your program must prompt the users for the exact same questions as the samples do. If your questions are in a different order you will lose marks from sections 3.1 as the autograder expects the questions in the same order. If your program phrases the questions differently, uses different spelling or punctuation you will lose marks from section 3.2 as you did not correctly follow the formatting guidelines. Etc. Remember we are looking for precise, exact, solutions.

```
Welcome to the CIS1500 Student Finance Planner. We have a few questions for you:

How much savings do you have set aside for the year? $ 15651.83

Total funds from Grants/Loans/Scholarships/Gifts? $ 2375.00

Do you have a part-time job? (Y or N) Y

        How much do you make per hour? $ 20.00

        How many hours do you work per week? 15

Do you receive an allowance? (Y or N) Y

        How often do you receive an allowance? (W for Weekly, M for Monthly, A for annually): M

        How much do you receive per allowance payment? $ 200.00

Do you pay for Housing? (Y or N) Y

        What is your monthly housing payment? $ 555.50

        Utilities cost per month? $ 76.99

How many semesters will you attend school this year? 2

What does tuition cost per semester? $ 4000.00

How much do you expect to pay for textbooks per semester? $ 0.00

Expenses:

        Enter a one-word name of an expense, if you are finished enter "finished": CarPayment

        What is the monthly cost of this expense? $ 400.00

        Enter a one-word name of an expense, if you are finished enter "finished": Insurance

        What is the monthly cost of this expense? $ 124.00

        Enter a one-word name of an expense, if you are finished enter "finished": Netflix

        What is the monthly cost of this expense? $ 10.00

        Enter a one-word name of an expense, if you are finished enter "finished": Food

        What is the monthly cost of this expense? $ 85.00

        Enter a one-word name of an expense, if you are finished enter "finished": Phone

        What is the monthly cost of this expense? $ 75.00

        Enter a one-word name of an expense, if you are finished enter "finished": finished
```

Your program will record values into variables and store them to be used in Part 2. We will not test for incorrect data types, ex asking for an integer and receiving a character.

2.3 Part 2: Generating a Finances Table:

Review the A1 outline for details on the printing the summary. The main differences will be:

- The code for generating the table should be moved to the appropriate function.
- You will require a loop to print the expenses arrays.
-

```
Finances Table:
=========================================              |
Funds Available                                        |
-----------------------------------------              |
        Funds at Start of Semester       | $ 15651.83   |
        Grants/Loans/Scholarships/Gifts  | $ 2375.00    |
        Total After Tuition and Books    | $ 10026.83   |
        Funds Per Month                  | $ 835.57     |
        Income Per Month                 | $ 1500.00    |
        Total Per Month                  | $ 2335.57    |
-----------------------------------------              |
Expenses Required (Per Month)                          |
-----------------------------------------              |
        Rent                             | $ 555.50     |
        Utilities                        | $ 76.99      |
        CarPayment                       | $ 400.00     |
        Insurance                        | $ 124.00     |
        Netflix                          | $ 10.00      |
        Food                             | $ 85.00      |
        Phone                            | $ 75.00      |
-----------------------------------------              |
Essentials Total                         | $ 1326.49    |
Essentials Available (50% of Funds)      | $ 1167.78    |
Remainder                                | $ -158.71    |
-----------------------------------------              |
Available for Wants                      | $ 700.67     |
-----------------------------------------              |
Available for Savings                    | $ 467.11     |
=========================================              |
```

2.4 Part 3: Display Summary

Review the A1 outline for details on the printing the summary. The main differences will that the code for printing the summary should be moved to the appropriate function. Remember

Note that wants and savings cannot be negative in the summary, if it is below 0 it should be set to 0 (Hint. the "fmax" function in math.h could help here). Ex. If a user's Funds, Grants, and income are less than tuition, the monthly funds available would be negative in the above table; you will need to ensure the value shows as a minimum of 0 in the summary (See Sample Flow Videos).

If we wanted to calculate the total growth of variable called yearlySavings with interest rate of 5% over 25 years, then we would **use the following interest equation**: $total = (yearlySavings * 1.05^{25})$. Hint: use the pow function in math.h.

```
You need to earn an additional $ 317.41 per month to pay your expenses and follow the 50/30/20 rule.

You have $ 700.67 per month to spend on "Wants" such as video games, nights out, hobbies, etc.

Over 1 year, you will set aside $ 5605.37 for savings.
In 5 years with 5% interest, your $ 5605.37 will have grown to $ 7154.03.
In 10 years with 5% interest, your $ 5605.37 will have grown to $ 9130.55.
In 25 years with 5% interest, your $ 5605.37 will have grown to $ 18981.76.
```

After this message your program will end.

Note: if your *essentials available* is $100 short of the *essentials required*, you will need to increase your *monthly funds available* by $200. This is because the essentials budget is ½ of the total budget.

3. Assessment

The assignment is graded out of 10 marks. The last page of this document shows an example of what a rubric/feedback file might look like. The 10 marks are split between these three categories:

3.1 The First 2 Marks

This section tests the overall flow of your program; your program will receive input from a user, do calculations, and output the table and summary. We will run multiple tests using different input values then compare your output against the expected output.

3.2 The Next 6 Marks

Function Testing: we will run multiple tests here to evaluate the required functions. To do this we will create a separate .c file with a new main, we will include your A2_Functions.c file, and call your functions with our main.

This will allow us to evaluate each function one at a time, so even if your program does not pass the flow tests, you can still receive marks for working functions.

It is imperative that you do not change the function headers from how they are shown in the prototypes in A2_Functions.h. The best option would be to directly copy from the .h file in the skeleton code, and paste into your A2_Functions.c. If you change the function headers in your .c file then errors will occur when we attempt to unit test the functions.

3.3 The Next 1 Marks

Your output text formatting will be graded out of 1. We will flag differences in spellings, spaces, tables, new lines, etc. and test against different input values using the auto-grader, so make sure you follow the output exactly. For each unique error, 0.5 marks will be deducted. The formatting grade will not be negative, it will only range from 0 to 1.

Refer to the sample flows provided for how your program is expected to run (4 Sample Flows). Altering/customizing sentences will not be accepted, and you will lose grades in this category if you do so.

3.4 The Final 1 Mark

Style: Indentation, variable names, and comments.

Comments are most commonly used to explain confusing or not easily understood parts of code or for depicting that the following code carries out a certain piece of logic. These are also used to explain the program and add a header to it. A file header comment is a type of comment that appears at the top of your program before the #include line(s). **Comment grades include header comments (see 5.3 File Header Comment Format) and meaningful comments throughout your code.**

Each code block should be **indented for readability** (if/else statements are an example of a code block). We are looking for **meaningful variable names** that depict the values they are representing – this will make it easier for you to remember what each variable is for and for us to grade you!

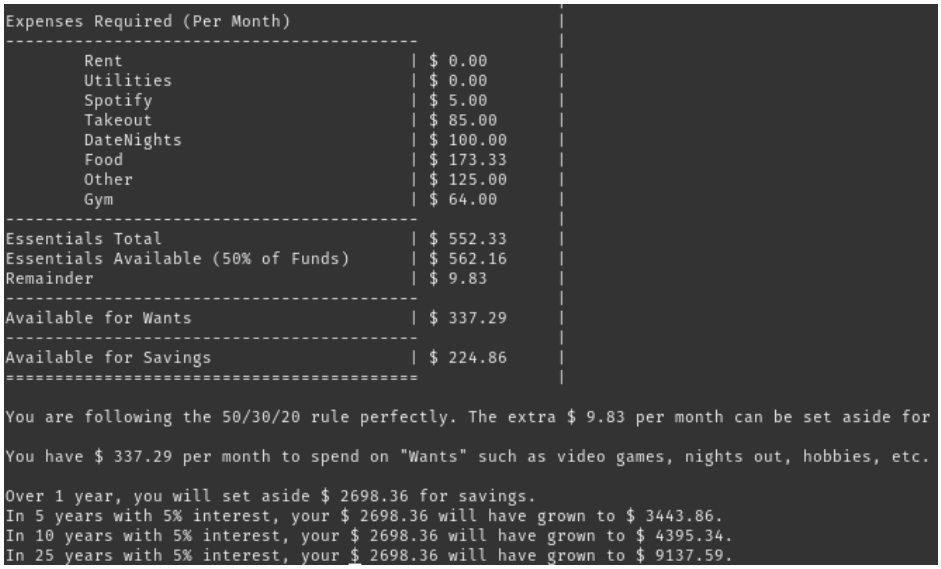
4. Sample Flows

Here is another example of the program being run. The skeleton code contains some of the table formatting already, so you can use that as a basis. Sub-questions are intended with a single tab (hint \t in printf).

Note that **your submission will be either fully or partially graded by an auto-grader**. To get full grades you need to match our output requirements exactly.

```
Welcome to the CIS1500 Student Finance Planner. We have a few questions for you:

How much savings do you have set aside for the year? $ 4501.64

Total funds from Grants/Loans/Scholarships/Gifts? $ 1000.00

Do you have a part-time job? (Y or N) Y

        How much do you make per hour? $ 20.15

        How many hours do you work per week? 15

Do you receive an allowance? (Y or N) N

Do you pay for Housing? (Y or N) n

How many semesters will you attend school this year? 3

What does tuition cost per semester? $ 2500.50

How much do you expect to pay for textbooks per semester? $ 75.12

Expenses:

        Enter a one-word name of an expense, if you are finished enter "finished": Spotify

        What is the monthly cost of this expense? $ 5.00

        Enter a one-word name of an expense, if you are finished enter "finished": Takeout

        What is the monthly cost of this expense? $ 85.00

        Enter a one-word name of an expense, if you are finished enter "finished": DateNights

        What is the monthly cost of this expense? $ 100.00

        Enter a one-word name of an expense, if you are finished enter "finished": Food

        What is the monthly cost of this expense? $ 173.33

        Enter a one-word name of an expense, if you are finished enter "finished": Other

        What is the monthly cost of this expense? $ 125.00

        Enter a one-word name of an expense, if you are finished enter "finished": Gym

        What is the monthly cost of this expense? $ 64.00

        Enter a one-word name of an expense, if you are finished enter "finished": finished

Finances Table:
========================================              |
Funds Available                                       |
----------------------------------------              |
        Funds at Start of Semester       | $ 4501.64    |
        Grants/Loans/Scholarships/Gifts  | $ 1000.00    |
        Total After Tuition and Books    | $ -2225.22   |
        Funds Per Month                  | $ -185.43    |
        Income Per Month                 | $ 1309.75    |
        Total Per Month                  | $ 1124.32    |
----------------------------------------              |
Expenses Required (Per Month)                         |
----------------------------------------              |
        Rent                             | $ 0.00       |
        Utilities                        | $ 0.00       |
        Spotify                          | $ 5.00       |
        Takeout                          | $ 85.00      |
        DateNights                       | $ 100.00     |
        Food                             | $ 173.33     |
        Other                            | $ 125.00     |
```

(Image continued from previous page.)



5. Program Submission and Administration Information

The following section outlines what is expected when submitting the assignment and various other administration information with respect to the assignment. To submit your assignment, upload a single zip file to the Dropbox box for A2 on Courselink.

5.1 The Submission File

The following is expected for the file that is to be submitted upon completing the assignment:

- You are to submit **one zip file containing two source code files** for your program.
- The name of the **zip file** follows the following format: **studentNumberA2.zip**
  - Example: John Snow's student number is 1770770 then the zip file name is 1770770A2.zip
- The two files within the zip you are to submit are non-compiled code C file (ends with .c) containing your code
  - Do not include any other files or folders in your submission; they will be ignored.
  - Do not include this file inside another folder; it will be ignored by the auto-grader.
- The name of the **C file containing main()** follows the following format: **studentNumberA2.c**
  - Example: John Snow's student number is 1770770 then the C file name is 1770770A2.c
- The name of the C file containing your functions is A2_Functions.c
  - Do not include the .h file
  - Do not change the function headers (return types, names, or arguments) from those in the .h file.
- **Incorrect zip file or C file name will result in a grade of 0.**
- To ensure you zip correctly and in the right format, we require you to zip your submission through the command line using the following command:

zip studentNumberA2.zip studentNumberA2.c A2_Functions.c

Make sure to replace studentNumber with your own student number.

5.2 Program Expectations

Your program is expected to follow the outlined information exactly. Failure to do so will result in deductions to your assignment grade.

- Your program should be compiled and tested on the SoCS Linux server.
- You must use the following command to compile your code:

  gcc -Wall -std=c99 studentNumberA2.c A2_Functions.c -lm

  - Example: For John Snow's submission, the command would be:
    gcc -Wall -std=c99 1770770A1.c A2_Functions.c -lm
- Programs that produce warnings upon compilation will receive a deduction of 1 mark for each type/category of warning
- The program file must contain instructions for the TA on how to compile and run your program in a file header comment (see 3.3 File Header Comment Format).

You will receive a 0 if:

- your program does not compile/creates errors when compiled with gcc -std=c99 -Wall on the SoCS Linux server.
- your program contains any global variables (variables declared outside of a function)
- you program uses:goto statements
- you program uses malloc
- you program uses calloc
- you program uses regex
- you program uses concepts not already taught in this course (ie. structs, dynamic arrays, etc.).

5.3 File Header Comment Format

Note: This sample uses the name John Snow; the **file name, student name, student ID, file descriptions, etc. must be changed**.

```
/************************ 1770770A2.c ************************

Student Name: John Snow                                    Student ID: 1770770

Due Date: Friday, 3rd  Nov 2023, at 11:59 pm

Course Name: CIS*1500

I have exclusive control over this submission via my password.

By including this statement in this header comment, I certify that:

1) I have read and understood the University policy on academic integrity; and

2) I have completed assigned video on academic integrity.

I assert that this work is my own. I have appropriately acknowledged any and all material (code, data, images, ideas or words) that I have used, whether directly quoted or paraphrase. Furthermore, I certify that this assignment was prepared by me specifically for this course.

*************************************************************

This file contains…

Describe the program, functions, important variables, etc.

*************************************************************

The program should be compiled using the following flags:

-std=c99

-Wall

-lm

Compiling:

gcc -Wall -std=c99 1770707A2.c A2_Functions.c -o A2 -lm

OR

Gcc -Wall -std=c99 1770707A2.c A2_Functions.c -lm

Running the program:

./A2

OR

./a.out

*************************************************************/
```

5.3 Grading Scheme:

This is an example of what feedback might look like. The 12 Function Test Cases will test the functions individually. We will use our own main function, and include your A2_Functions.c file, then test each function individually. For example, our program might have the code:

```
float unitTest1 = getFloatInput("Unit test1: ");

printf("%f\n", unitTest1);
```

And then enter -10 and 42. We expect that "Unit test1: " is printed, then the phrase "Invalid Input" is printed, followed by the "42". This way we can test each function not just the complete program. The 2 flow tests will be similar to those in a1, looking at the overall flow of the program given a complete set of finance inputs.

A2 Marking Scheme: 1770770

Style (out of 1.0)

* Indentation (0.5/0.5)
* Commenting/Readability (0.5/0.5)

Test Cases and Outputs (out of 8.0)

* Function Test Case 1 (1.0/1.0)
* Function Test Case 2 (1.0/1.0)
* Function Test Case 3 (0.5/1.0)
* Function Test Case 4 (1.0/1.0)
* Function Test Case 5 (1.0/1.0)
* Function Test Case 6 (0.0/1.0)
* Function Test Case 8 (1.0/1.0)
* Function Test Case 9 (1.0/1.0)
* Function Test Case 10 (0.5/1.0)
* Function Test Case 11 (1.0/1.0)
* Function Test Case 12 (1.0/1.0)
* Flow Test 1 (0.0/1.0)
* Flow Test 2 (0.0/1.0)

Grade for Output Formatting (1.0/1.0)

Final grade: 8.5 /10.0

TA Comments:

Should say "Welcome to the CIS1500 Student Finance Planner. We have a few questions for you:" not "Hello and Welcome". Program did not exit after invalid input from use.