Team #9 Members:
Kimberly Price (Team Leader)
Kaji Rashad
Andrew Chabab
Jamal West
Alexander Gonzalez
Alberto Molina

Professor: Dr. Jonathan C.L. Liu
Class: COP3503 (Programming Fundamentals II)
Date: 16th April 2015

Team Project Milestone#3 (Report)

The team project assignment in the course of "COP3503: Programming Fundamentals II" is very crucial. It allows the students (programmers) to intermingle with other students to create a usable source code that can be beneficial in the real world. It also allows the students to experiment with software engineering and get an in-depth understanding of it. Software engineering refers "to the study of software development on large scales." (Dr. Liu, Lecture 16: Slide 2) For our scenario, we created a team consisting of six members. Initially, we had multiple meetings dedicated towards actually coming up with the idea for the project and defining our C++ based software system. Finally, we were able to come up with the idea of creating a small C++ application that would be labeled "Schedule Optimizer".

The system definition was as follows: our "Schedule Optimizer" program will make an optimal academic schedule based on user preferences, class needs and available times. Our system is important because it would save users' time when deciding on their semester schedules and it is easily applicable and useful in the real world.

The source code was built to intake the following input from the user to run the program smoothly:

- Course object: Course name (string), course number (string), section (string array), times available (string array), and credits (int).

- Preferences object: Minimum and maximum credits (int), days off (bool), maximum period in a row (int), maximum period in a day (int), and unwanted periods (string).
- Schedule object: Course object(s).
- Comparison function: Schedule object(s) and preference object(s).

After taking in these inputs, the program was designed to output matrices (mxn) to the console, displaying possible academic schedules for the user. The rows (m) are the possible periods (time) and the columns (n) are the possible days i.e. Monday, Tuesday, Wednesday, etc. The final schedule can also be exported to a comma separated file (which can be opened with programs such as Excel for it to be shown in its proper format) and/or text file. The schedule will have different views for the user such as weekly planner view (comma separated file) and course list view (text file). The program will take all of the parameters into consideration and use our algorithm to create several optimal schedules for the user. These will be printed three at a time and the user will have the choice to print three more or select one in particular to save as a comma separated and/or text file.

The flow and logic of the program can be summarized as follows: the user will input the required parameters about the course information and their schedule preferences. Then the method “addClass” will use these parameters to come up with all the possible combinations of courses. Once we have all the combinations, another method “bestSched” will compare these schedules with the user’s preferences and delete the ones that do not adhere to these. Another method, “displaySched”, will print out the combinations that match the user’s preferences three at a time. Lastly, the user will decide if he or she wants to see more variations of the schedule or print a particular one into a file. (Note: If all preferences are not able to be met, the program will display the closet match and list out the preferences that were not met.)

In our program, we wanted to model the real world more accurately by implementing a very strict input process and allowing the user to actually input course name and course number of real classes such as: Calculus/ MAC2311 or Physics/PHY2049. Our program is designed under the assumption that the user will follow the guidelines for inputting. However, input validation is implemented to prevent improper input to be used. The style of the output is simple enough that no user should have a problem understanding it.

In our system, a simple modification can modify the whole schedule. Therefore, we implemented a confirmation method in our source code which ask the user if the information he or she inputted is accurate every time they input information for the schedule. If not, we incorporated functions to take in minor modifications and alter the user's initial inputs with the given modifications.

To be very specific and detailed, our team has decided to elaborate upon each class and functions associated with it. To begin with, we will start with the class "Section". Section has three private data fields: string sectionNum, int days [10] and int periods [10]. The array index is interchangeable and can be modified upon the user's request. The class "Section" has the following public constructor: "void set (string sectionNum, string day[], string period[])" which intakes the given arguments and assigns them accordingly. For example, it assigns the inputted section numbers(s) to the string variable "sectionNum". It also initializes the string day and period array accordingly to the sectionNum to represent the meeting days and time periods for that section. To add on, the periods are assigned accordingly and converted into "int" using the C++ predefined function "atoi". Within this class, there were setter and getter methods to make the implementation of the class simple and accessible by other classes.

Moving on, for the class "Course", these are the following private data fields: string courseName, string courseNum, Section section[10] and int credits. Course has the following public constructors: "void set(string courseName, string courseNum, Section section[], int credits)" which intakes the given arguments and assign them accordingly. Within this class, there are also getters and setters which makes the implementation of the class easy and accessible by other classes.

After these two classes, the rest are functions which utilize these functions and their data fields as references to implement the rest of the source code. We first start with the function "void processTimeAvbl" which converts the user input for time availability to two arrays of days and corresponding periods. Within this function, we first check to see if the user is inputting valid inputs by checking the first index of the section time availability array and making sure it's a letter. If not, we ask the user to try again. Similarly, we make sure the second index is a number which represents the period in which the section is held. We associated the implementation of the function "int stoi(string input, bool &wrongInput)" to cast string to int and throw an

exception if there is a wrong input. It also confirms if the input is correct by referencing the wrongInput variable.

After the implementation of all of these functions, we move onto one of our most crucial functions labeled “void getCourseInfo (Course course[])” which has the following data fields: string courseName, string courseNum, string sect, string priority, string time, string line and string in. Additionally, there are some Boolean variables initialized to assist with input validations. To summarize, this function helps incorporate all of the previous classes and function. For example, this function helps ask the user for the course name, course number, course credits, number of sections associated with each course and time availability for each section. As stated before, the initialized boolean values are used for input validations.

Meanwhile, in terms of the class “Preferences”, these are the following private data fields: int minCredits, int maxCredits, bool anyDaysOff, int maxPeriodRow, int maxPeriodDay and string unwantedPeriod. The function “void getPrefs()” assist in retrieving information from the user such as: the minimum number of credit hours, the maximum number of credit hours and whether the user wants any days off. In addition, there is input validation incorporated to make sure the user is following the proper input guidelines. After the user passes that phase, they are asked for more data: the maximum number of periods they want in a row and the maximum number of periods they want in a day. We also took into consideration whether the user wants to modify something after inputting all of the information; we were able to address this issue by creating a boolean variable called “confirmed”. Finally, there is a “printPrefs” function which prints the user inputs and allows them to observe the information to make sure it is accurate one last time. As always, there are getters and setters which help in the implementation of the class and makes it accessible to other classes.

In the class “Schedule” we have the following data fields: string classSched[14][5], bool isOccupied[14][5], string incl[5] and stirng notIncl[5]. The data field “string classSched[14][5]” creates an array of schedule which is plotted over 14 rows and 5 columns. The 5 columns represent the days of the week (Monday, Tuesday, Wednesday, Thursday and Friday). The 14 rows represent the 14 periods available at the University of Florida. Next, we have the data field “bool isOccupied[14][5]” to see which slot within the plot of 14 rows and 5 columns is occupied. We can use this information to avoid plotting a class onto spot that is occupied. The data field “string incl[5]” and “string notIncl[5]” is used to see which courses were included onto the

schedule and which courses were not. Following these data fields, there is initializing of the following variables to 0: numCredits, numCourse and attemptCourses. After the initializing, the default "Schedule()" constructor is defined. The default constructor allows us to set all of the spots within the plot to "NULL" and set the boolean for isOccupied for all of the plots to "false". Following right after this constructor, there is a "void addClass(Course course, int section)" which helps in plotting each class onto the class schedule grid. It also has a validation concept incorporated which checks whether it is plotting a class onto a spot which is already occupied. The function "void displaySched()" prints the schedule onto the console. Last but not least, there is a function "bestSched(Preferences pref, Schedule schedArray[])" which intakes the arguments from class preference/function pref and class schedule/function schedArray[] and compares both in order to output the best optimized schedule which is the sole purpose of our code.

We split up our source code into the following components: Schedule, FileIO, Preferences, Course, and Algorithm.

- The "Schedule" component's task was to implement a source code which would add classes onto the correct slot of the schedule grid and make sure each spot only contains one class. In addition, another function of this component was to display the schedule onto the console.
- The "FileIO" component's task was to implement code which would output the results in a Planner View (Comma Separated File) and List View (Text File).
- The "Preferences" component's task was to intake the user's preferences in terms of the minimum and maximum number of credits he or she would like to pursue after, the maximum number of periods he or she would like to have in a row and day and whether they wanted days off.
- The "Course" component was responsible for handling and storing any information regarding the course such as course name, course number, number of sections involved with the course, and the times of each section.
- Last but not least, the algorithm component was responsible for putting all of the other components together and unifying them.

From this project, we were able to learn a plethora of lessons. First and foremost, communication is the main key in software engineering. Each team member needs to know the status of the other person's progress and how they are implementing their portion of code

because in a sense, each portion depends on the other portions or simply put: the main unified algorithm is dependent. This dependency causes each team member to need to implement his or her code to make it compatible with the rest of the code. In addition, another lesson learned was that there is more time for debugging the sooner the testing process is started. It is wise to start the testing process a month before the due date of the project which gives plenty of time for debugging. However, at times, the debugging process can take longer if the issue is very complex. Therefore, start the testing process as soon as possible. To add on, dividing up the task among the team members is crucial. For example, assigning a team member to work on a portion he or she isn't good at can truly hurt the team. Therefore, it is always wise to divide up the work accordingly to each member's strength rather than weakness, this way everyone can add the most and learn from the others on the areas they are not as proficient in. Last but not least, showing up to meetings on time and being organized is another lesson learned. The team will be able to move at a faster pace because everyone is on the same page. In addition, the team members will be able to feed off each other's thoughts and progress during the weekly meetings. On that note, it is also wise to hold weekly meetings until the due date of the project assigned. Conducting weekly meetings versus bi-monthly or monthly meetings help the members who miss any meeting to be less clueless because they only missed one week's worth of reflection compared to two or four weeks of reflection.

In terms of future work, there are many more features we could incorporate in the source code. We could incorporate a function which intakes a repository from the University's registrar office that contains all of the available courses information such as: course name, course number, number of sections per each course, and meeting times for each section. By incorporating this function, the code would depict the real world more accurately and be more resourceful. We could use the "Course" component of our current source code and compare the user's input with the given registrar office's file from the university. After the comparison, the outputted schedule would be authentic and can actually be put to use. Also, we could make the selection process easier and give the user more options in terms of outputting multiple schedules. In terms of online classes, we could include a function which takes online classes that do not have any meeting times. For example, as of now, in our code, the user must enter a meeting time for each section. We don't have a function which would actually intake a "NULL" entry and consider that input as an online website based class. Another direction the team wanted to head to originally

was having a class called “Student” which would store the student’s information (UFID, date of birth, major and minor (if applicable) and courses already taken). By having a student class, a student can return to the program and continue from where he or she left off from. It would allow the user the option of continuing to work on their schedule throughout the day and week rather than having to input all of the preferred classes in one compile time run. We could also implement more user input validation to make the input process stricter. The number of periods in a day can be modified depending upon the user’s choice of university and the number of periods available each day at that university so the experience can be tailored to students all over the country. Moreover, the team thought it would be beneficial to have another function which would allow the user to edit a schedule after it’s printed onto the console. The user will be able to add or remove a course. Additionally, another function could be implemented which would allow the user to import a schedule from a file (that was made using our program) instead of having to re-input course information. Last but not least, our team recognized that some schedules would be printed more than once due to the nature of our algorithm. Therefore, to conclude, our team would’ve liked to incorporate a function which would prevent the same schedule from printing out more than once to the console.

Overall, this team project was a great experience for every team member involved to grow and learn from. It helped us experience software engineering at a deeper level and reflect upon it. In the real word of software engineering, a programmer should be comfortable working with others. This team project assignment helped some team members reflect upon the experience to see if they really enjoy working in teams compared to working on their own. All in all, our team is satisfied with the amount of work we were able to accomplish and incorporate in our program within the given deadline.

Example Run #1

```
Enter your first name: Kaji
Enter your last name: Rashad
Enter your UFID(no dashes): 61516285

Would you like to edit your information?

First name: Kaji
Last name: Rashad
UFID: 61516285
Yes(1) or No(0): 0


Course Information
Please enter the courses of interest in order of priority.
How many courses would you like to add?
5
Enter course name: Bio

Enter course number: BSC2010

Enter number of credits: 3

How many sections of Bio would you like to add? 3

Enter section number: 1234

Enter class availability in the following form:
M,T,W,R or F for Monday,Tuesday,...,Friday respectively
followed by the period number. Separate each period with commas
in the following manner T6,R6,R7: M3,W3,F3

Enter section number: 4567

Enter class availability in the following form:
M,T,W,R or F for Monday,Tuesday,...,Friday respectively
followed by the period number. Separate each period with commas
in the following manner T6,R6,R7: T6,T7,R6
```

This picture shows the input process from the user. It initially asks the user to input his or her name and UFID. Afterwards, it asks the user if he or she wants to edit any information then it moves onto collecting the course information from the user.

```
Enter section number: 8910

Enter class availability in the following form:
M,T,W,R or F for Monday,Tuesday,...,Friday respectively
followed by the period number. Separate each period with commas
in the following manner T6,R6,R7: W6,W7,W8

Information for course Bio
Course number: BSC2010
Credits: 3
    - Section 1234 at times: M3,W3,F3
    - Section 4567 at times: T6,T7,R6
    - Section 8910 at times: W6,W7,W8

Enter 1 to continue or 2 to edit this information: 1

Enter course name: Chem

Enter course number: CHM2045

Enter number of credits: 4

How many sections of Chem would you like to add? 3

Enter section number: 2345

Enter class availability in the following form:
M,T,W,R or F for Monday,Tuesday,...,Friday respectively
followed by the period number. Separate each period with commas
in the following manner T6,R6,R7: T3,R3,R4

Enter section number: 6789

Enter class availability in the following form:
M,T,W,R or F for Monday,Tuesday,...,Friday respectively
followed by the period number. Separate each period with commas
in the following manner T6,R6,R7: M5,W5,F5
```

This is the continuation of the input process from the previous picture. The program is continuing to collect information about the user's courses.

```
Enter section number: 1011

Enter class availability in the following form:
M,T,W,R or F for Monday,Tuesday,...,Friday respectively
followed by the period number. Separate each period with commas
in the following manner T6,R6,R7: F6,F7,F8

Information for course Chem
Course number: CHM2045
Credits: 4
    - Section 2345 at times: T3,R3,R4
    - Section 6789 at times: M5,W5,F5
    - Section 1011 at times: F6,F7,F8

Enter 1 to continue or 2 to edit this information: 1

Enter course name: Calc

Enter course number: MAC2011

Enter number of credits: 4

How many sections of Calc would you like to add? 3

Enter section number: 3456

Enter class availability in the following form:
M,T,W,R or F for Monday,Tuesday,...,Friday respectively
followed by the period number. Separate each period with commas
in the following manner T6,R6,R7: M5,W5,F5

Enter section number: 78910

Enter class availability in the following form:
M,T,W,R or F for Monday,Tuesday,...,Friday respectively
followed by the period number. Separate each period with commas
in the following manner T6,R6,R7: M4,W4,F4
```

This is the continuation of the input process from the previous picture. The program is continuing to collect information about the user's courses.

```
Enter section number: 7834

Enter class availability in the following form:
M,T,W,R or F for Monday,Tuesday,...,Friday respectively
followed by the period number. Separate each period with commas
in the following manner T6,R6,R7: R7,R8,R9

Information for course Calc
Course number: MAC2011
Credits: 4
    - Section 3456 at times: M5,W5,F5
    - Section 78910 at times: M4,W4,F4
    - Section 7834 at times: R7,R8,R9

Enter 1 to continue or 2 to edit this information: 1

Enter course name: English

Enter course number: ENC1101

Enter number of credits: 3

How many sections of English  would you like to add? 3

Enter section number: 7894

Enter class availability in the following form:
M,T,W,R or F for Monday,Tuesday,...,Friday respectively
followed by the period number. Separate each period with commas
in the following manner T6,R6,R7: R7,R8,R9

Enter section number: 7431

Enter class availability in the following form:
M,T,W,R or F for Monday,Tuesday,...,Friday respectively
followed by the period number. Separate each period with commas
in the following manner T6,R6,R7: M4,W4,F4
```

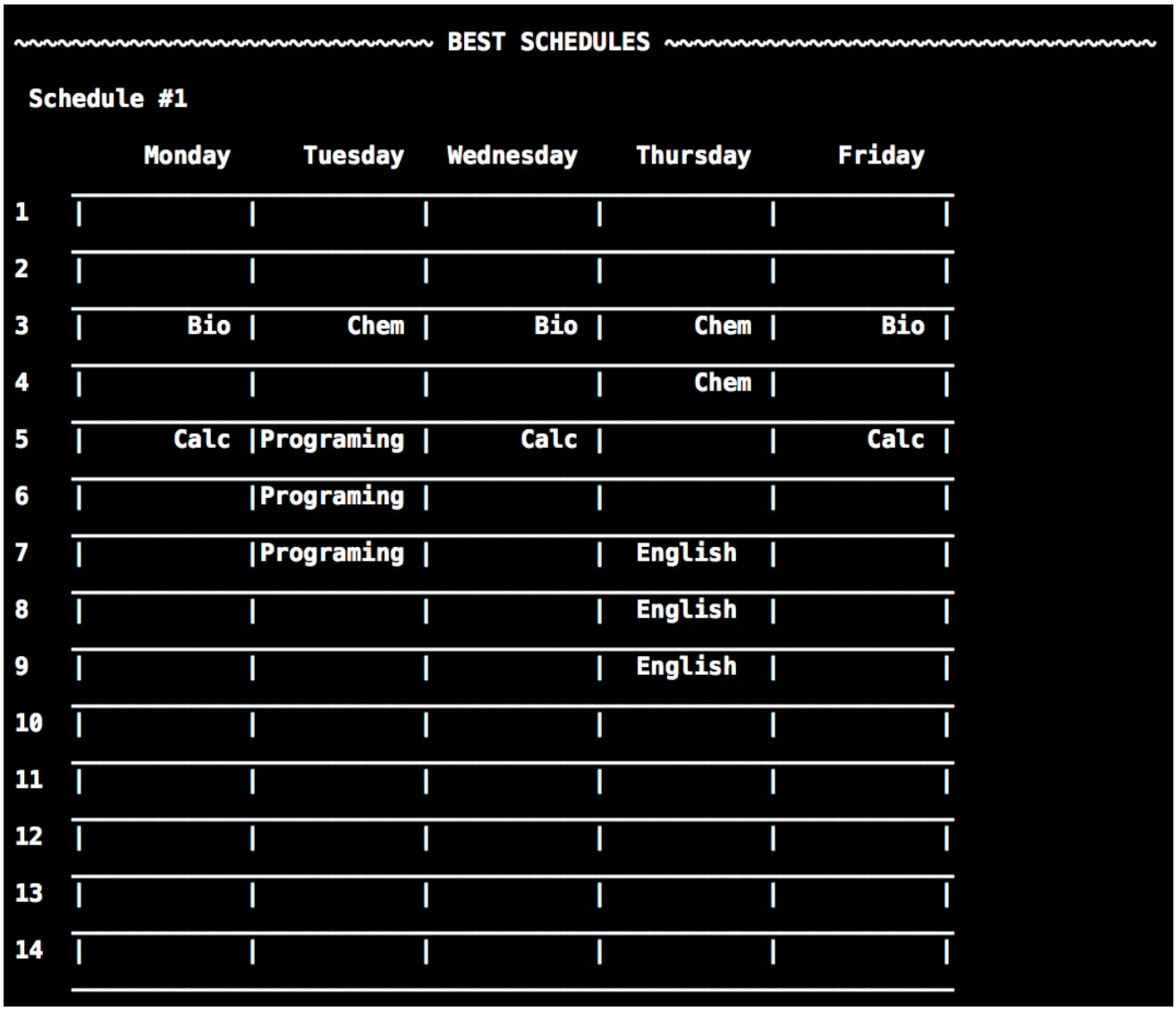
This is the continuation of the input process from the previous picture. The program is continuing to collect information about the user's courses.

```
Enter section number: 95422

Enter class availability in the following form:
M,T,W,R or F for Monday,Tuesday,...,Friday respectively
followed by the period number. Separate each period with commas
in the following manner T6,R6,R7: W9,W10,W11

Information for course English
Course number: ENC1101
Credits: 3
    - Section 7894 at times: R7,R8,R9
    - Section 7431 at times: M4,W4,F4
    - Section 95422 at times: W9,W10,W11

Enter 1 to continue or 2 to edit this information: 1

Enter course name: Programing 2

Enter course number: COP3502

Enter number of credits: 3

How many sections of Programing 2 would you like to add? 3

Enter section number: 1111

Enter class availability in the following form:
M,T,W,R or F for Monday,Tuesday,...,Friday respectively
followed by the period number. Separate each period with commas
in the following manner T6,R6,R7: T5,T6,T7

Enter section number: 2222

Enter class availability in the following form:
M,T,W,R or F for Monday,Tuesday,...,Friday respectively
followed by the period number. Separate each period with commas
in the following manner T6,R6,R7: M1,M2,M3
```

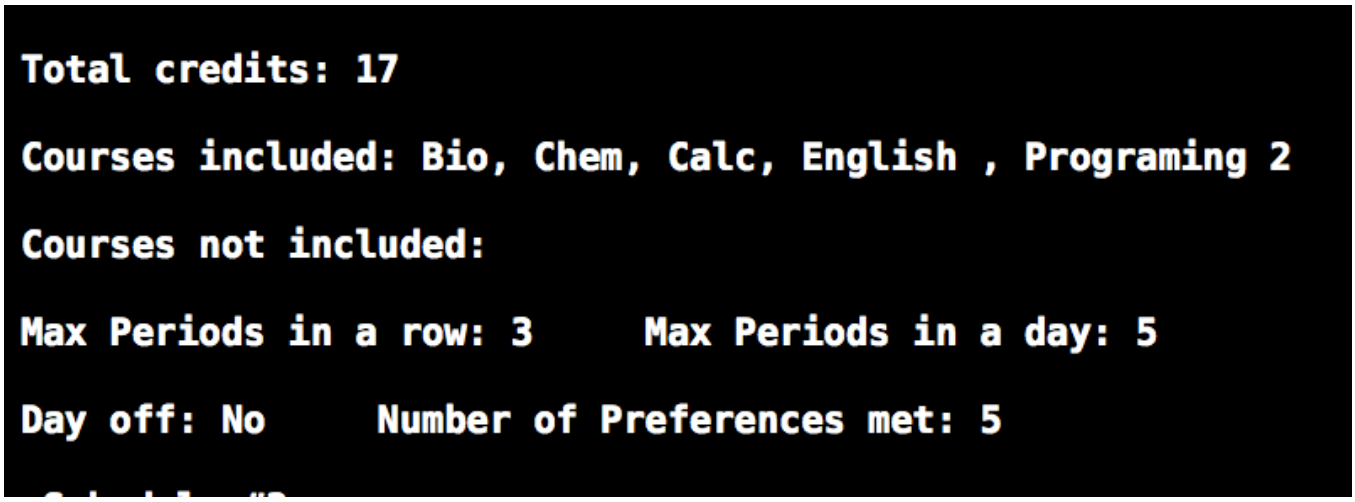
This is the continuation of the input process from the previous picture. The program is continuing to collect information about the user's courses.

```
Enter section number: 3333

Enter class availability in the following form:
M,T,W,R or F for Monday,Tuesday,...,Friday respectively
followed by the period number. Separate each period with commas
in the following manner T6,R6,R7: F1,F2,F3

Information for course Programing 2
Course number: COP3502
Credits: 3
    - Section 1111 at times: T5,T6,T7
    - Section 2222 at times: M1,M2,M3
    - Section 3333 at times: F1,F2,F3

Enter 1 to continue or 2 to edit this information: 1


Schedule Preferences
What is the minimum number of credit hours you prefer for your Schedule? 10
What is the maximum number of credit hours you prefer for your Schedule? 18
Would you like any days off? (Y)/(N) N
What is the maximum number of periods you prefer in a row? 5
What is the maximum number of periods your prefer in a day? 5

------- Schedule Preferences -------
(1) Minimum Credits: 10
(2) Maximum Credits: 18
(3) Requested Days Off: NO
(4) Maximum Periods in a Row: 5
(5) Maximum Periods in a Day: 5
Would you like to change anything in your Preferences? (Y)/(N) N
```

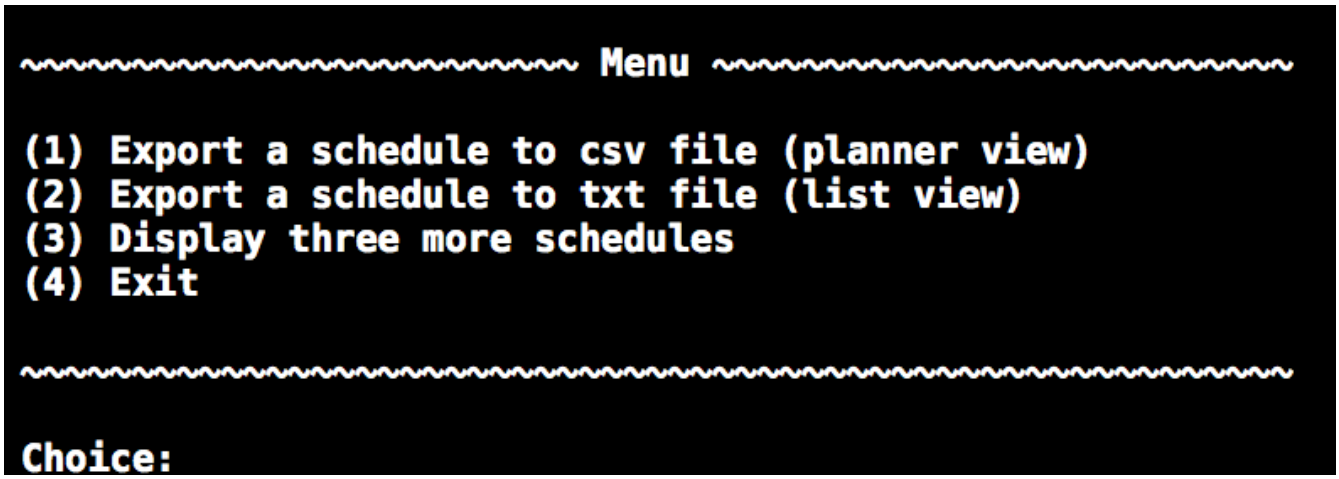
This is the continuation of the input process from the previous picture. The program is continuing to collect information about the user's courses. In addition, after the input process is done in terms of collecting information about the user's courses, we move onto the preferences and ask the user to input his or her preferences as shown above.



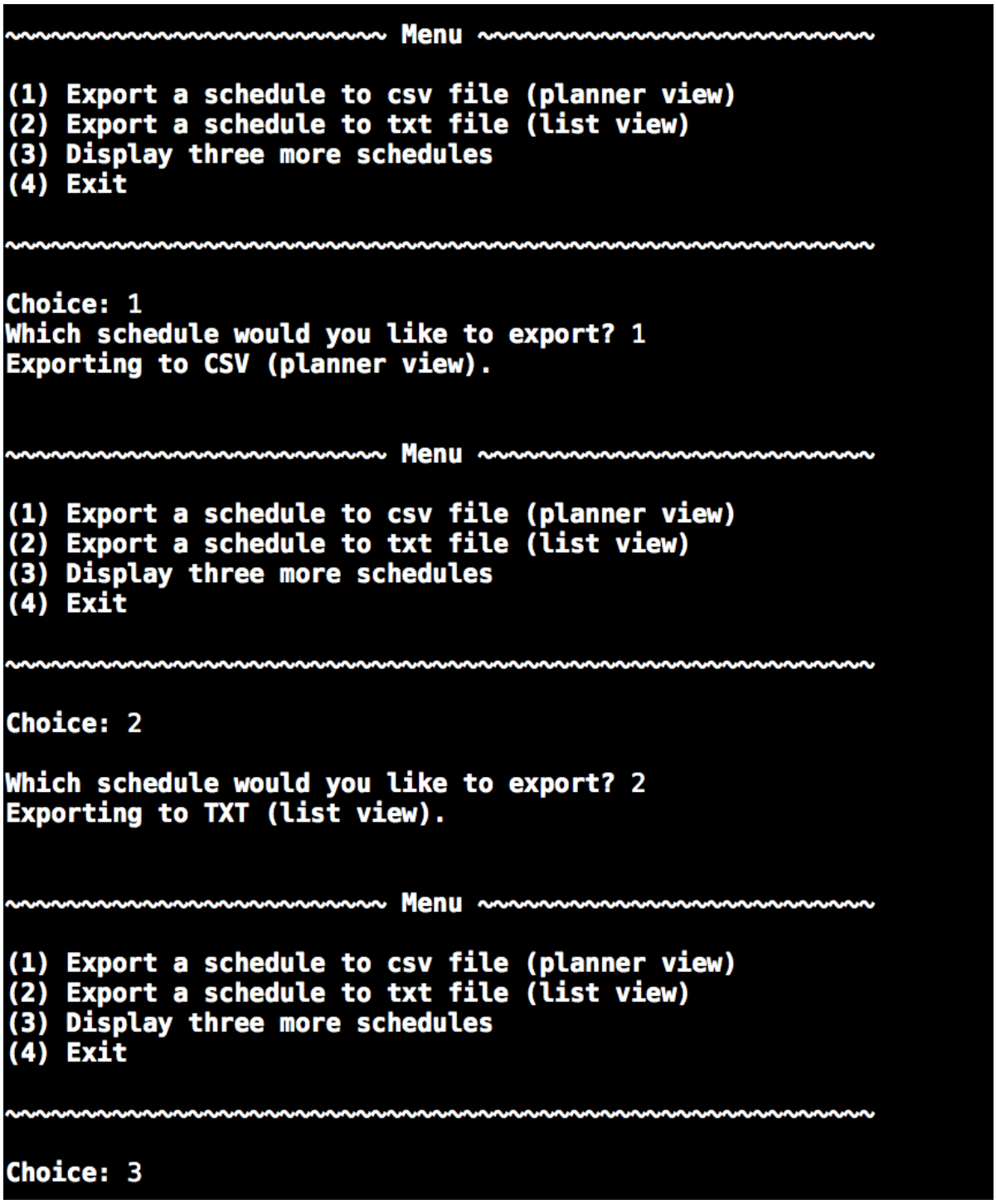
This is the first schedule outputted to the console that meets the highest number of preferences. There are also two more schedules followed by this schedule.



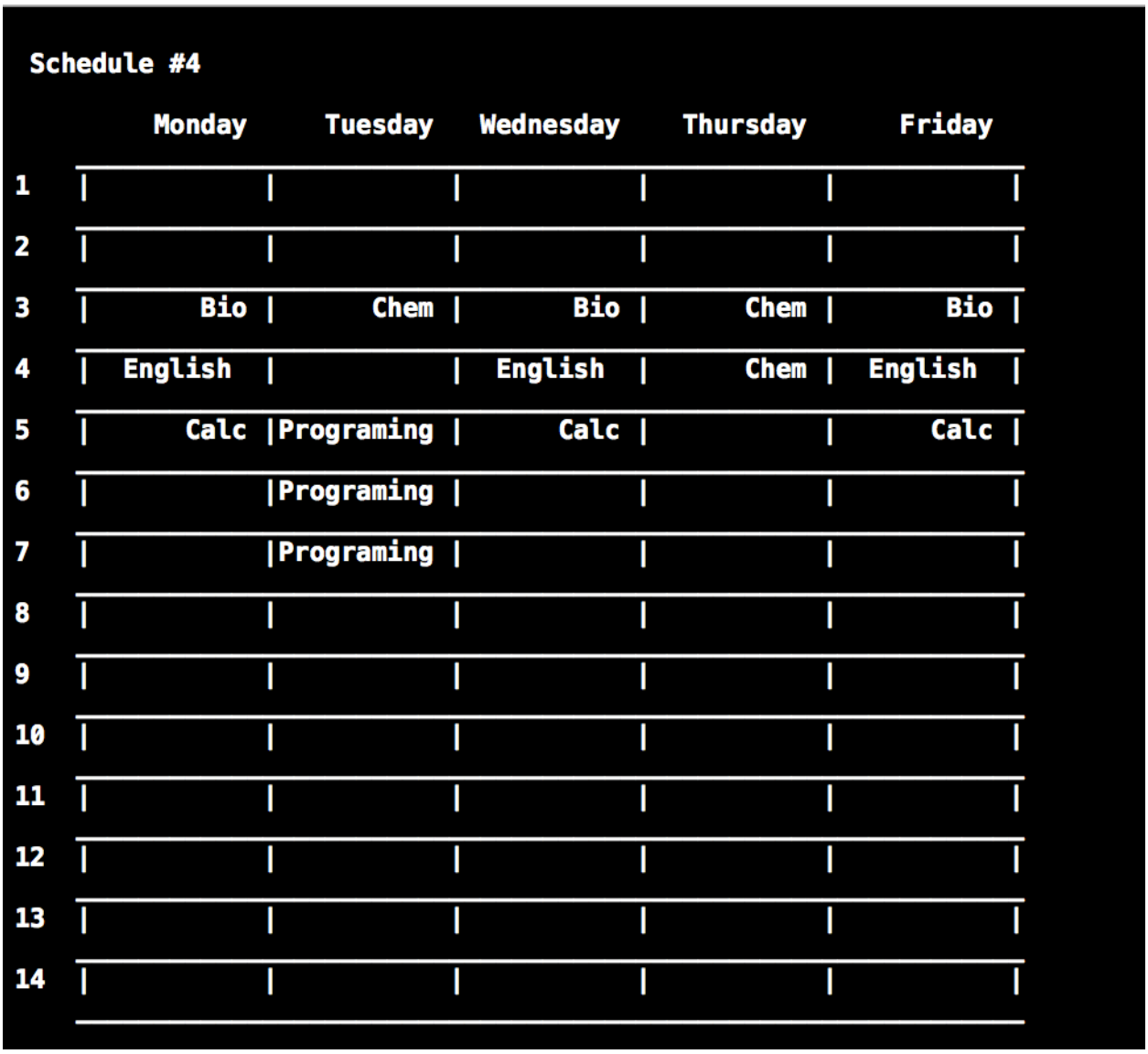
Right after the schedule is printed, the program displays information relevant to the schedule. First and foremost, it displays how many accumulated credits the schedule contains. Moreover, it showcases the classes it was able to plot onto the schedule and the courses that were not plotted onto the schedule hence “courses not included”. In addition, it displays the maximum periods in a row and maximum periods in a day. Right after, it displays whether the schedule has a day off, and the number of preferences met (5 in this example run).



This is the menu printed to the console right after the schedules are printed. It gives the user the options to export his or her schedule into a csv file (planner view) or txt file (list view). In addition, it lets the user display three more schedules if he or she wants too. Last but not least, it allows the user to exit the program.



Inputting choices from the menu. At first, we export the first schedule into a csv file and then the same schedule into a txt file. Afterwards, we display three more schedules.



This is what is shown to the console when the user decides to print 3 more schedules (choice 3). Schedule four is followed by schedule five and six.

```
UFID: 61516285
Dear Kaji Rashad,
Here are your classes for the semester in an easy-to-follow list view! Enjoy!

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
Course: Bio
        Course number: BSC2010
        Course section number: 1234
        Course meeting times: M3,W3,F3
        Course credits: 3
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

Course: Chem
        Course number: CHM2045
        Course section number: 2345
        Course meeting times: T3,R3,R4
        Course credits: 4
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

Course: Calc
        Course number: MAC2011
        Course section number: 3456
        Course meeting times: M5,W5,F5
        Course credits: 4
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

Course: English
        Course number: ENC1101
        Course section number: 7894
        Course meeting times: R7,R8,R9
        Course credits: 3
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
```

This is the exported txt file (list view) for the user. It contains all of the courses plotted onto the first schedule. It also contains course name, course number, course section number, course meeting times, and course credits.

	A	B	C	D	E	F	G
1	Periods	Monday	Tuesday	Wednesday	Thursday	Friday	
2	1						
3	2						
4	3	Bio	Chem	Bio	Chem	Bio	
5	4				Chem		
6	5	Calc	Programing 2	Calc		Calc	
7	6		Programing 2				
8	7		Programing 2		English		
9	8				English		
10	9				English		
11	10						
12	11						
13	12						
14	13						
15	14						
16							
17							

This is the exported csv file (Excel file) which displays the schedule. It plots the course names onto the grid, showing the specific period(s) and day(s) that each course meets.