**CASE STUDY: COPY-TYPING PROJECT SCHEDULING SOFTWARE**

A Project Presented to the

College of Computer Studies - Computer Science Department

FEU Institute of Technology

In Partial Fulfillment

of the Requirements for the Software Engineering

by

**John Mheynard Azuela**

**John Marklenard Macatangay**

**Smith Ramoy**

8 January 2018

1. Introduction

Copy typing jobs are in demand worldwide especially to those internet users seeking for work from home or online jobs. As a copy typist, they receive documents that people wanted to be converted into an editable Word documents (Virtu@dmin, 2017). In order to be hired as a copy typist, one must simply know how to type multiple documents fast. Since a lot of documents are being handled by a copy typist, time is being considered to accomplish the tasks. Most of the tasks cannot be done in one sitting. Therefore, a schedule for copy typing projects is introduced. The purpose of this project is to provide an order and flow in doing such copy typing projects, to achieve efficiency, and to determine how well the work is being done. The project is somehow related with our course because it was one of our prior knowledge in operating systems. This project involves First Come, First Serve (FCFS) scheduling or First In, First Out (FIFO) in which the tasks are being processed first in the order that they arrive in queue. Another scheduling is being applied is priority scheduling. It is the most common scheduling algorithm in which each job has its own priority. The job with the highest priority (lowest numerical value like 1) is to be executed first. These scheduling algorithms help copy typists to manage their work efficiently.

1. Data Structures and Algorithms

This software involves data structures and algorithms that makes the required functions to work. For the implementation of data structures in this software, one of them is queue. Queue or First In, First Out is a data structure type in which the element inserted first, will be removed first as well (Sehgal,2017). This data structure type is suitable because data don't stay as long as with stacks. It is also fast and flexible unlike the other data structures such as stacks. For the algorithms part of this software, one of them is the bubble sort algorithm. Bubble sort algorithm is an iterative process of comparing each element with another and sorting them depending upon the given values (Sehgal,2018) . The algorithm starts with first two elements, determining which is greater than the other in order to sort to increasing order. Same goes to another set of pairs in an unsorted list of elements. This kind of algorithm is known as brute force approach (Anon,2013). This algorithm is used in the software rather than other sorting algorithms because it takes less memory space and is easy to implement. For the searching algorithm of this software, linear search is implemented. Linear search is the most basic type of searching algorithm in which it searches for an element in a list sequentially without skipping (Sehgal,2017). It is also in a form of a brute force method. With a row of elements, linear search is being done by attempting to find a match from left to right. This algorithm is used in the software rather than other searching algorithms because just like bubble sort, it is simple, easy to understand, and takes less memory space.

1. The Program Functionalities

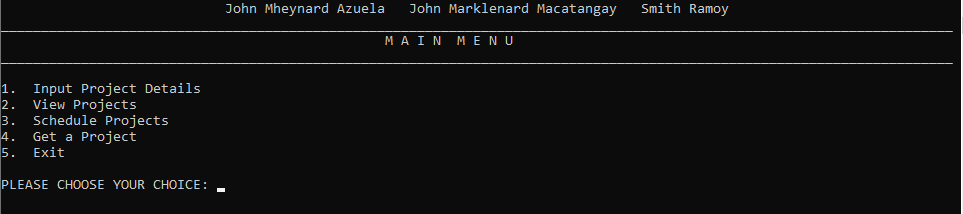


Figure 3.1: Main Menu of the Software

Starting off with the components of the software main menu, it is composed of the following: (a) Input Project Details, (b) View Projects, (c) Schedule Projects, (d) Get a Project, (e) Exit, and (f) header. This is what it looks like once this part of the software is converted to codes.

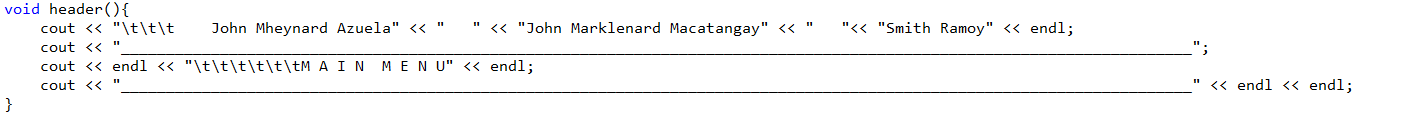


Figure 3.2 Code snippet of the software header

This particular code snippet is for the header which uses void header(). This function contains group members’ names at the top of the prompt, the word “MAIN MENU”, and the two horizontal lines compose of underscores.

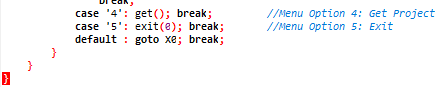
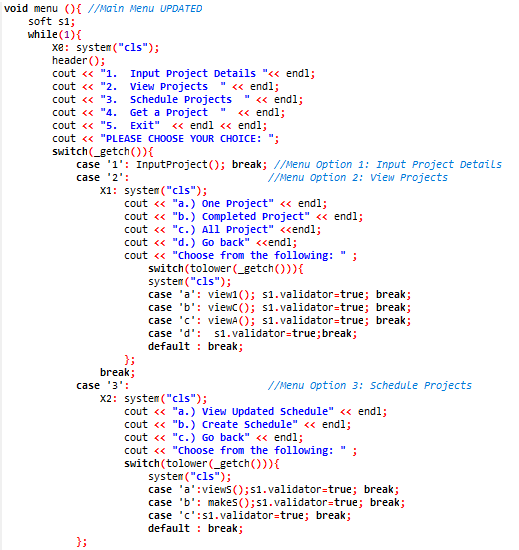


Figure 3.3 Code snippet of the software main menu

This code snippet is for the software main menu which uses void menu() function. Header() is being called to display the software header. System(“cls) allows to clear the screen of the prompt. Once the prompt is clear, the main menu appears. To choose from the five options, \_getch() function is used within the switch-case statements. \_getch() function allows to input a single character without pressing Enter. The main menu options only accept integers ranging from 1 to 5. Another bunch of switch(“cls) are used to clear the screen in order to display new prompts.

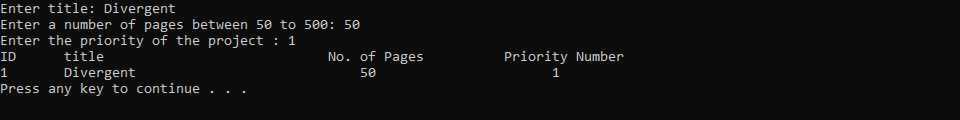


Figure 3.4 Input Project Details

Once a user inputs “1” on their keyboard, the program prompts to input the title, number of pages within 50 to 500, priority number of the project. The program outputs what the users input. Once it’s done, they can press any key to return back to the main menu. This is what it looks like once it is converted to codes.

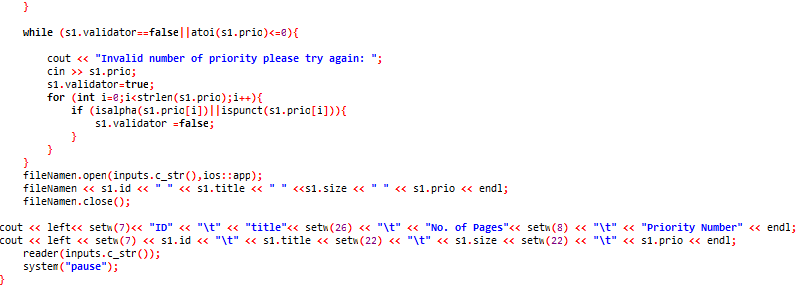
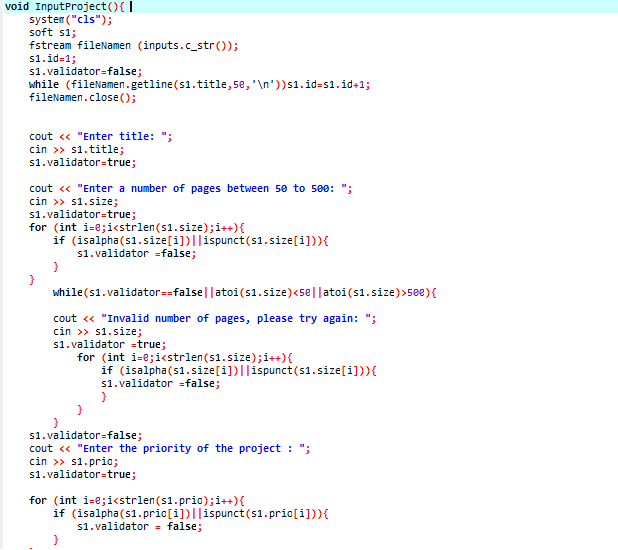


Figure 3.5 Code snippet of Input Project Details (Menu Option 1)

This code snippet shows that void InputProject() is used. System(“cls”) allows to clear screen once the user inputs “1” on their keyboard. For the number of pages, a user must input not less than 50 and greater than 500. Negative numbers are considered to be invalid. Also, they aren’t allowed to input non-numerical characters such as letters and symbols. Same goes for the priority number of the project, negative numbers, letters, and symbols are invalid. The ID number is in integer and it’s automatically assigned to a certain project. It always increments by 1.



Figure 3.6 View Projects (Menu Option 2)

Once a user inputs “2” on their keyboard, three options appear on the prompt. These are (a) One Project, (b) Completed Project, and (c) All Project. The program only allows to input letters a, b, and c.

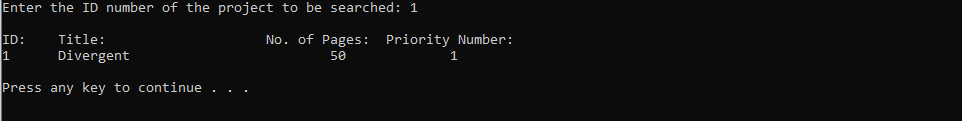


Figure 3.7 One Project

Once a user inputs ‘a’ on their keyboard, the program prompts to input the ID number. If the ID number along with project details exists, it will be shown on the prompt. Otherwise, no results found. Afterwards, pressing any key leads them back to the main menu. This is what it looks like once it is converted to codes.



Figure 3.8 Code Snippet of One Project

This particular code snippet shows that void view1() is used. System(“cls”) allows to clear screen once the user inputs ‘a’ on their keyboard. For the project ID number, the program only accepts positive integers. Letters and symbols are invalid. The program searches for an existing ID number in Input Project.txt file.

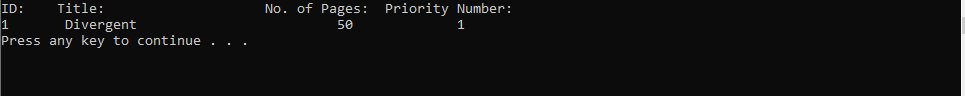


Figure 3.9 All Project

Once a user inputs ‘c’ on the keyboard, the prompt displays all the projects whether they are completed or uncompleted. Pressing any key leads back to the main menu. This is what it looks like once it is written in codes.

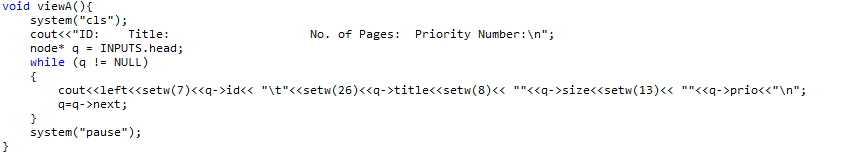
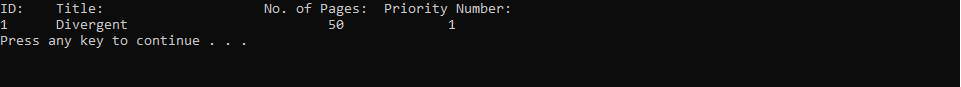


Figure 3.10 Code snippet of All Project

This code snippet shows that void view(A) is used. The variable q allows to search on the Input Project.txt. Setw() uses to set width for output operations. Pointer and while loops are being used within the function.



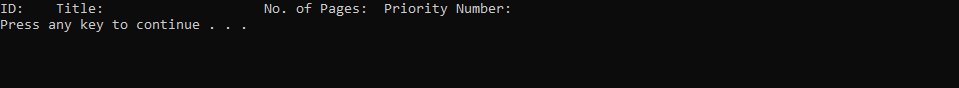


Figure 3.11 Completed Project

Once a user inputs ‘b’ on the keyboard, the program display list of completed projects. The list could be empty because Completed Project.txt is empty as well. Pressing any key leads back to the main menu. This is what looks like once it is written in codes.

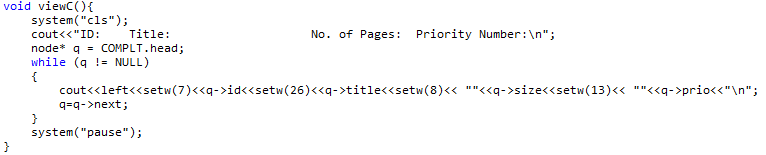


Figure 3.12 Code Snippet of Completed Project

This particular code snippet shows void viewC() is used. The variable q allows to search on the Completed Project.txt. Setw() uses to set width for output operations. Pointer and while loops are being used within the function.



Figure 3.13 Schedule Project (Menu Option 3)

Once a user inputs “3” on the keyboard, the prompt displays another options to choose from (a) View Updated Schedule and (b) Create Schedule. This part of the software only accepts letters a and b.

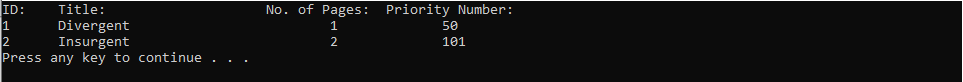
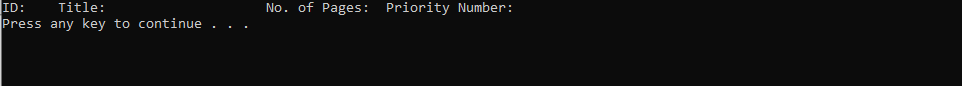
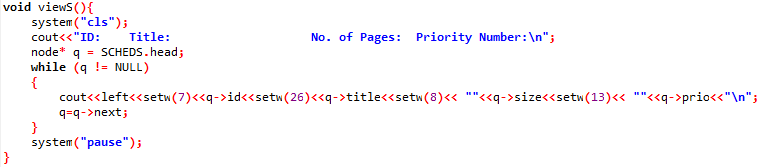


Figure 3.14 View Updated Schedule

Once a user inputs ‘a’ on the keyboard, the prompt displays the created schedule of projects. They are arranged according to their priority numbers. If priority numbers of the projects are the same, the page numbers would be considered as another basis in sorting. This is what it looks like once it is written in codes.

Figure 3.15 Code Snippet of View Updated Schedule



This particular code snippet shows void viewS() is used. The variable q allows to search on the Sched Project.txt. Setw() uses to set width for output operations. Pointer and while loops are being used within the function.

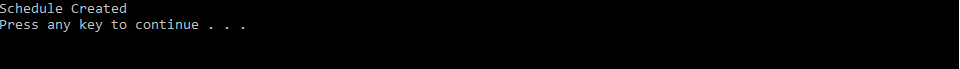


Figure 3.16 Create Schedule

Once a user inputs ‘b’ on their keyboard, a “Schedule Created” message on the prompt appears. Pressing any key leads back to main menu. This is what it looks like once it’s written in codes.

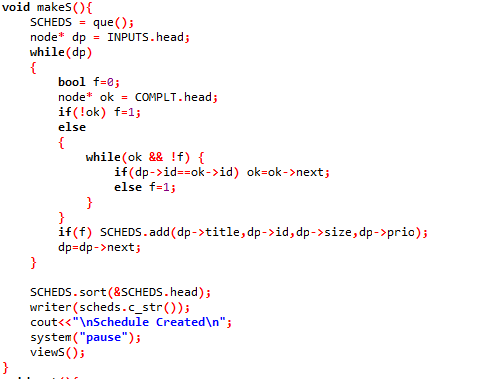


Figure 3.17 Code Snippet of Create Schedule

This particular code snippet shows that void makeS() is used for creating schedule. SCHEDS.add concatenates the project details and SCHEDS.sort sorts the projects in increasing order based their priority numbers.



Figure 3.18 Get a Project (Menu Option 4)

Once a user inputs “4” on their keyboard, the program gets the project from updated schedule. “Project [title] has been completed.” message appears. It means to say that this project will be placed under Completed Projects. Pressing any key leads back to main menu. This is what it looks like when it’s written in codes.

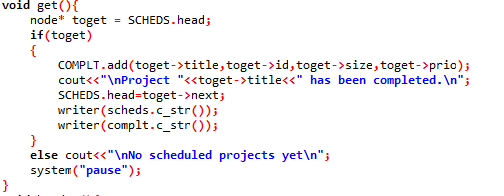


Figure 3.9 Code Snippet of Get a Project

This code snippet shows that void get() function is used in getting the project. The variable toget takes the first project in a list and exports the details in Completed Project.txt

1. Tests and Validation

This section shows the validation of every function in this program.

• **MAIN MENU**

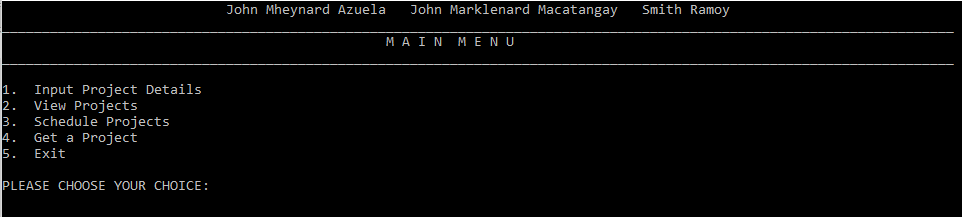
****

Figure 4.1 If a user inputs any letters, symbols, and numbers that aren’t in the choices, the program wouldn’t give an error. The program works only if the right input is entered.

**• Input Project Details**



Figure 4.2 The only error that exists in the program is once a user enters a project title with spaces, it messes up the program.

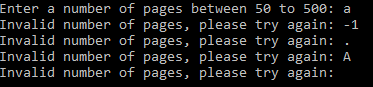


Figure 4.3 A user attempts to input upper and lowercase letters, negative number and a symbol. The result is an error.

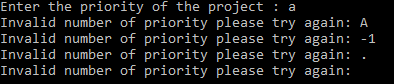


Figure 4.4 A user attempts to input upper and lowercase letters, negative number and a symbol. The result is an error.

**• View Project**

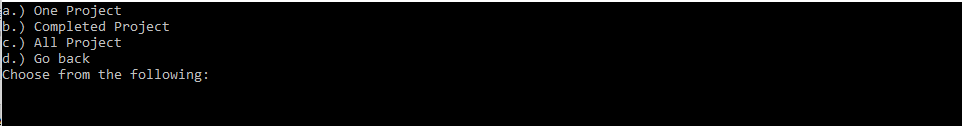


Figure 4.5 A user attempts to input letters that aren’t in the choices, numbers and symbols. The program doesn’t give any errors, but the program isn’t working unless right input is entered.

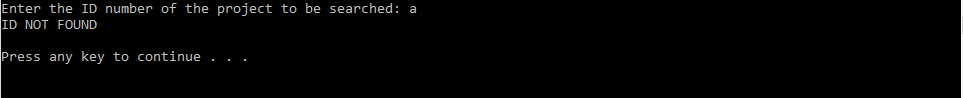


Figure 4.6 A user attempts to enter non existing ID number.

Figure 4.6 A user attempts to enter lowercase letter.

Figure 4.6 A user attempts to enter uppercase letter.

Figure 4.6 A user attempts to enter a symbol.

Figure 4.6 A user chooses Completed Project even Completed Project.txt is empty

**• Get a Project**

V. Conclusion and Recommendation

Copy typists consider time and effort to accomplish such tons of documents. With this project, this can aid to accomplish such requirements with a great pace in a manner of speaking time.

For the ease of usage of this software, better visuals of the interface is recommended. Another reason why a better visual interface is recommended because most people like attractive displays to avoid boredom when using the software. A simple design and color scheme are appropriate for software’s graphical user interface as long as the words are readable and don’t hurt people’s eyes.

In creating a project, time and patience is needed. The expectations of the project were met. Everything run smoothly as expected. The only problem that was left behind is the spaces between the title. It will not show the whole title when it is viewed. To make this project better, have great knowledge in file handling, and the different structures in C++. It will also lessen the time needed to complete the project. Looking back at the project, it is simple for the user to easily understand the flow of the program.

VI. Reference List

Anon., (2013). *Bubble Sorting ~ codefreak.* [Online]

Available at: https://codingfreak.blogspot.com/2013/12/bubble-sorting.html

[Accessed 1 January 2019].

Sehgal, K. (2017). *A Simplified Explanation about Data Structures*. [online] Available at: https://medium.com/karuna-sehgal/a-simplifed-expanation-about-data-structures-ddaddd209737 [Accessed January 1, 2019].

Sehgal, K. (2017). *A Simplified Explanation about Linear Search*. [online] Available at: https://medium.com/karuna-sehgal/an-simplified-explanation-of-linear-search-5056942ba965 [Accessed January 1, 2019].

Sehgal, K. (2018). *An Introduction to Bubble Sort*. [online] Available at: https://medium.com/karuna-sehgal/an-introduction-to-bubble-sort-d85273acfcd8 [Accessed January 1, 2019].

Virtu@dmin, (2017). *What does a copy typist do and what is copy typing?.* [Online]

Available at: https://medium.com/@cyberspaceayuda/what-does-a-copy-typist-do-and-what-is-copy-typing-379ce17a4678

[Accessed 1 January 2019].