

DE LA SALLE UNIVERSITY - MANILA

SOS: Student Organizer Software (Prioritization To-do list)


A Term Project

Presented to Mr. Ramon Stephen L. Ruiz


In Partial Fulfillment of the

Requirements for the Course Programming Logic and Design (PROLOGI)

by

BARJA, Samuelle P.  S.P.B

CHUA, Hazel Melody  H.M.C

UY, Kae T.  K.T.U

EQ3

Monday-Thursday

April, 2023

Table of Contents

Table of Contents	2
List of Tables and Figures	3
I. Introduction	4
Background of the Study	4
Problem statement	5
Objectives	5
Significance of the Project	6
II. Review of Related Literature	8
III. Methodology	11
V. Discussion of Results	38
VI. Analysis, Conclusion, and Future Directives	39
References	40
Appendices	42
A. User's Manual	42
B. Source Code –	47
C. Work breakdown	53
D. Personal Data Sheet	56

List of Tables and Figures

Figure 1	14
Figure 2	15
Figure 3	16
Figure 4	16
Figure 5	17
Figure 6	17
Figure 7	29
Figure 8	29
Figure 9	30
Figure 10	31
Figure 11	31
Figure 12	32
Figure 13	32
Figure 14	32
Figure 15	33
Figure 16	33
Figure 17	34
Figure 18	34
Figure 19	35
Figure 20	35
Figure 21	35
Figure 22	36
Figure 23	36
Figure 24	37
Figure 25	37
Figure 26	37
Figure 27	42
Figure 28	43
Figure 29	43
Figure 30	44
Figure 31	44
Figure 32	45
Figure 33	45
Table 1	11
Table 2	53

I. Introduction

Background of the Study

The increasing workload and busy schedules of students can often lead to stress and anxiety, making it difficult to manage their time efficiently. To address this issue, the Student Organizer Software (SOS) has been developed. SOS is a program that helps students to organize and prioritize their tasks effectively. Unlike other to-do list organizers, SOS utilizes an algorithm that considers the urgency and importance of each task, based on Eisenhower's Urgent/Important principle (Indeed, 2022). This principle categorizes tasks into four quadrants based on their level of urgency and importance, allowing students to identify which tasks require immediate attention and which ones can be addressed later.

Another unique feature of SOS is that it takes into consideration the difficulty level of each task. This feature is especially useful for students who struggle with managing their workload and often feel overwhelmed by the tasks at hand. By considering both the urgency and importance of each task, as well as its difficulty level, SOS creates a prioritized to-do list that can help students stay organized and focused.

SOS will be developed using the C-language and will prompt the user to input the specific task and its due date. The program will also ask the user to rate the urgency and importance of each task based on a given criteria list, and to provide the task's difficulty level on a scale of 1-4. Based on these inputs, the program will generate a prioritized to-do list that will help the user to manage their workload more efficiently.

In summary, the Student Organizer Software (SOS) is a unique program that aims to assist students in managing their workload more effectively. By considering both the urgency and importance of each task, as well as its difficulty level, SOS can help students prioritize their tasks and stay on track with their assignments.

Problem statement

Procrastination and poor time management continue to be a major problem for students, leading to negative impacts on academic performance and mental health. According to a recent study, an estimated 80%-95% of college students procrastinate (Shatz, 2023). With the increasing use of technology and external distractions, managing one's time has become even more challenging for students. This problem is further exacerbated by the sudden increase in workload that students face, especially as face-to-face classes slowly resume.

The negative effects of poor time management and procrastination on students' academic performance and mental health are well-documented in the literature. Studies have shown that poor time management leads to increased stress, anxiety, and poor academic performance (Klassen, Krawchuk, & Rajani, 2008). Moreover, students who procrastinate tend to experience lower levels of well-being and higher levels of anxiety and depression (Sirois & Pychyl, 2013).

Given the pervasive nature of this problem, our team has decided to create a project that aims to contribute to abolishing procrastination and promoting good time management habits among students. By developing the Student Organizer Software (SOS), our goal is to provide students with a tool that can help them manage their time more effectively, prioritize their tasks, and reduce the negative effects of procrastination on their academic performance and mental health.

Objectives

The research objectives of this project are:

1. To design and develop a prioritization to-do list software program that considers both the urgency and importance of tasks and their difficulty level.
2. To evaluate the effectiveness of the Student Organizer Software (SOS) in helping students to organize their weekly tasks and showcase their deadlines.
3. To determine the impact of the SOS program on students' time management and productivity levels.
4. To assess the effectiveness of Eisenhower's Urgency/Importance principle in prioritizing tasks within the SOS program.

5. To identify any potential issues or limitations with the SOS program and suggest possible improvements to enhance its effectiveness in promoting good time management habits.

By achieving these objectives, this project aims to provide students with an effective tool for managing their time and prioritizing their tasks, ultimately promoting better academic performance, and reducing stress levels associated with procrastination and poor time management.

Significance of the Project

The Student Organizer Software (SOS) has the potential to provide a practical solution to the ongoing problem of procrastination and poor time management among students. The software's unique features, including the consideration of both the urgency and importance of tasks and their difficulty level, can help students to better manage their workload, reduce stress levels, and improve their academic performance.

The significance of this project lies in its potential to provide students with a tool that can help them to prioritize their tasks and stay on track with their assignments, ultimately leading to better academic outcomes. Additionally, by evaluating the effectiveness of the SOS program, this project can provide valuable insights into the impact of time management tools on students' productivity and well-being.

The negative effects of poor time management and procrastination on students' mental health and academic performance are well-documented in the literature. Therefore, by developing a program that can help students to overcome these challenges, this project has the potential to make a significant contribution to the field of education.

Moreover, by assessing the effectiveness of the Eisenhower Urgency/Importance principle in prioritizing tasks within the SOS program, this project can provide insights into the applicability of this principle in real-world situations.

In summary, this project's significance lies in its potential to provide students with an effective tool for managing their time and prioritizing their tasks, ultimately promoting better academic

performance, and reducing stress levels associated with procrastination and poor time management.

II. Review of Related Literature

In this section, a review of literature related to the project will be provided. The literature review is to provide background information and a strong foundation for the overall project implementation.

Importance of time management

With the gradual increase in workload, many people have more or less experienced inefficiencies, such as insomnia caused by anxiety or excessive work pressure. Thus, it is necessary to have a healthy time management to let not just students but also people who have a lot of work to do to better manage their time.

According to the survey conducted by Alyami et al. (2021) about the Effect of Time Management on Academic Achievement of Students. The majority of students believe that pre planning strategies are effective for enhancing academic achievement, as indicated by the 64.8% who agreed or strongly agreed with this statement. In contrast, 37.3% of respondents said they manage their time. 92.3 percent of the students said they met their deadlines.

According to the result of research conducted by Correia (2021) , As people have less stress at work, their productivity increases as well. Good time management will help reduce stress at work. Thus, this demonstrates how effective time management can help employees perform things at work more efficiently. In addition, successful time management might inspire people to accomplish activities more effectively. Once an individual organizes their to-do plan, it will motivate them to complete the tasks on the to-do list. When tasks on the to-do lists are completed, people will feel a feeling of accomplishment and will be more motivated to finish the remaining tasks as stated by Parenthetical (2020).

Moreover, Having good time management skills might also benefit in getting more sleep. When people get less than seven hours of sleep, they may become overworked, which can affect their memory and attention, slow down their reactions, and make them take longer to complete tasks. More extreme results could increase one's likelihood of developing conditions including anxiety, irritability, heart disease, obesity, and many more (Sleep Foundation, 2023). Aside from this, the research also stated that in the last two weeks, exhaustion at work affected about 38% of

workers. This shows that many people endure sleep deprivation as a result of poor time management, which makes many people feel exhausted at work.

Negative Effects of Procrastination

Procrastination is a widespread phenomenon that has negative consequences for individuals and society as a whole. In this review of related literature, we will examine the negative effects of procrastination on academic performance, mental health, and physical health.

One of the most note-worthy negative effects of procrastination is its impact on academic performance. According to Steel and Klingsieck (2016), students who procrastinate tend to have lower grades and academic achievement. This negative effect can be attributed to the fact that procrastination often leads to last-minute cramming and reduced quality of work (Steel & Klingsieck, 2016).

In addition to its impact on academic performance, procrastination has also been linked to mental health issues such as anxiety and depression. Sirois and Pychyl (2013) found that individuals who procrastinate are more likely to experience negative emotions and have lower well-being. This negative effect can be attributed to the fact that procrastination creates a cycle of stress and avoidance, which can lead to feelings of guilt and anxiety (Sirois & Pychyl, 2013).

Finally, procrastination can also have negative consequences for physical health. According to Tice and Baumeister (2018), individuals who procrastinate are more likely to engage in unhealthy behaviors such as poor sleep habits, lack of exercise, and unhealthy eating habits. This negative effect can be attributed to the fact that procrastination often leads to a lack of time management and poor decision-making (Tice & Baumeister, 2018).

In conclusion, procrastination has negative effects on academic performance, mental health, and physical health. Students, professionals, and individuals in general should be aware of these consequences and take steps to overcome procrastination to prevent these negative outcomes.

Eisenhower's Urgent/Important principle

Eisenhower's Urgent/Important principle is a time management technique that has gained popularity in recent years. According to this principle, tasks can be categorized into four quadrants based on their urgency and importance. This principle is attributed to former US President Dwight D. Eisenhower, who was known for his ability to manage his time effectively.

The first quadrant consists of tasks that are both urgent and important. These tasks require immediate attention and should be given top priority. Examples of such tasks include important deadlines and emergencies (Covey, 1989).

The second quadrant consists of tasks that are important but not urgent. These tasks should be given priority but can be scheduled for later. Examples of such tasks include long-term planning, personal development, and relationship building (Covey, 1989).

The third quadrant consists of tasks that are urgent but not important. These tasks should be delegated or postponed. Examples of such tasks include phone calls, emails, and interruptions (Covey, 1989).

The fourth quadrant consists of tasks that are neither urgent nor important. These tasks should be eliminated or minimized as they do not contribute to the achievement of one's goals (Covey, 1989).

Several studies have supported the effectiveness of Eisenhower's Urgent/Important principle in improving time management and productivity (Claessens, et al., 2014; Gutiérrez, et al., 2020; Song & Lee, 2019). These studies suggest that individuals who use this principle are better able to prioritize their tasks, reduce stress, and achieve their goals.

In conclusion, Eisenhower's Urgent/Important principle is a useful time management technique that can help individuals prioritize their tasks based on their urgency and importance. By using this principle, individuals can increase their productivity, reduce stress, and achieve their goals.

III. Methodology

In this section, we will delve into the methodology of our project and explore the various tools and techniques used to achieve our objectives. We will discuss the hierarchy chart, flowchart, and pseudocode, which are essential components of our development process. These visual aids help to organize and structure our ideas, allowing us to better understand the logic and flow of our program. By explaining each of these components in detail, we hope to provide a clear understanding of our methodology and how we approach the development of our project.

a. Conceptual Framework - IPO chart

Table 1

INPUT	PROCESS	OUTPUT
name[100] difficulty importance due_date choice pick	num_tasks = 0 day[7]={7,6,5,4,3,2,1} Construct a call a function for each options looping the code when num_tasks is more than one Constructing a for loop when the the inputted variable is greater than 1 priority = (difficulty * importance) / day[due_date-1] Constructing inner loop for priority Select case for choice : prioritization order , due- date weekly calendar , or exit	prioritization order priority due-date weekly calendar name due_date

	Select case for Storing the tasks under the appropriate day of the week using if statement to make decisions on the priority repeating the progress if choice = 'y' or 'Y'	
--	--	--

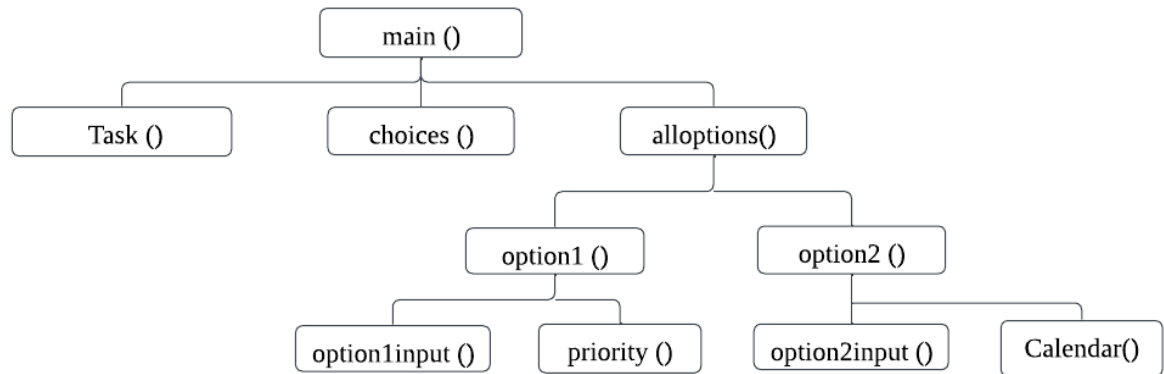
In these program, the first thing to do is to construct the “call a function” for all of the options, for each of the “call a function”, we need to first declare the variables, The variables that have been declared are the variables that are inputted in the INPUT and OUTPUT bar. The first “call a function” is for the users to choose their choices, 1 for the prioritization order and 2 for the due-date weekly calendar, and 3 for exiting the program. Then the second “call a function” will be containing the content for the prioritization order, First, it will be asking the user to input the number of tasks that the user wants to input using the “num_tasks” variable. If the inputted value is more than 1, the for loop is applied for repeating the content inside there. In addition, inside the “call a function” there’s still two “call a function” inside the body. The first nested function is the input of option 1 which is the prioritization order. Inside the function, it will be asking the user to input the name, difficulty, importance , and the due_date of the task. The variable “name” is applied for getting the name, “difficulty” for knowing how difficult the task is with 1 for the most difficult and 4 for the least difficulty. Next is the “importance” variable with the same rating as “difficult”. Furthermore, the variable “due_date” is used to know when the due_date of the task. Then next, it will be automatically calculating the priority of the task by using the formula “priority = (difficulty * importance) / day[due_date-1]”.Next is the second nested function, which is used for arranging the priority of the task input which also has made use of the for loop also. The next function is the body of the second option due-date weekly calendar, Then it will also be asking first the task that the user wants to input inside the variable “num_tasks”. Then 2 nested functions have been made. The first nested function is the one with the content of the second option which is used for knowing the name and the due_date of the task. Then for the second nested function which is used for Keeping tasks in the week array under the proper day of the week. Finally for

the last loop, a conditional statement has occurred, This function also has 2 nested functions inside the content of the body. The first nested function is used for calling the option1 body, and the second nested function is used for the second option, and case 3 is used for exit.

After constructing the codes, we can now proceed to the main body of the program. First, we also need to declare the variables that will be used. Then a “do-while loop” is used if there’s a person inputting “y” or “Y” inside the variable pick. Then next is just to call the function that was needed then we can now exit the program.

b. Hierarchy Chart

Figure 1



This code has a hierarchy structure with the main function at the top, followed by three modules/tasks, choices, and alloptions. The alloptions module has two subsections, option1 and option2. Option1 has two additional subsections, option1input, and Priority, while option2 has one more subsection called Calendar.

The choices module is responsible for printing out the menu options of the program. Option1 collects the task's name, difficulty, importance, and due date from the user via option1input, and then assigns each task a priority value based on its input data through the Priority module. The tasks are then printed in ascending order of priority. On the other hand, option2 collects the task's name and due date only from the user through option2input and assigns each task to the appropriate day of the week through the Calendar module.

In summary, the code allows the user to input task details and either arrange them by priority or assign them to their respective day in a week.

c. Flowchart

Figure 2

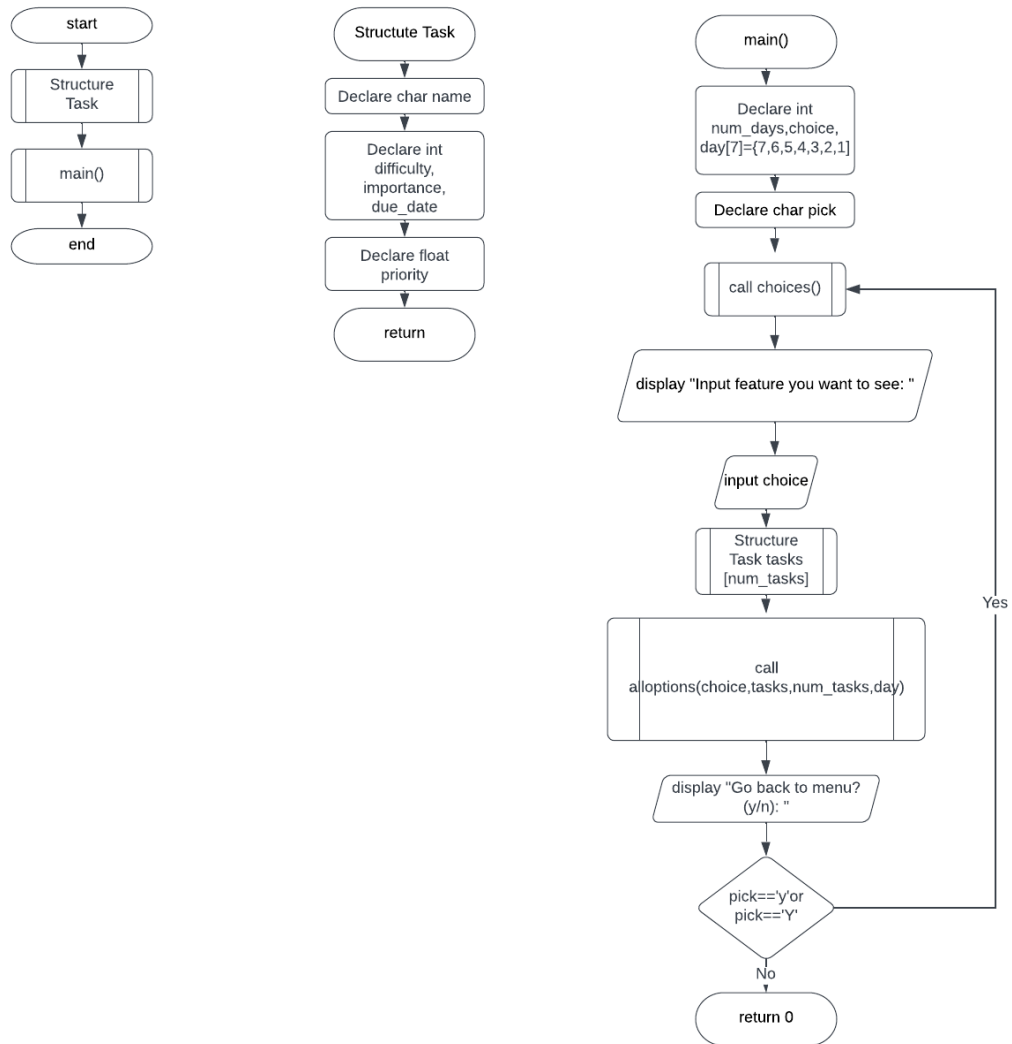


Figure 3

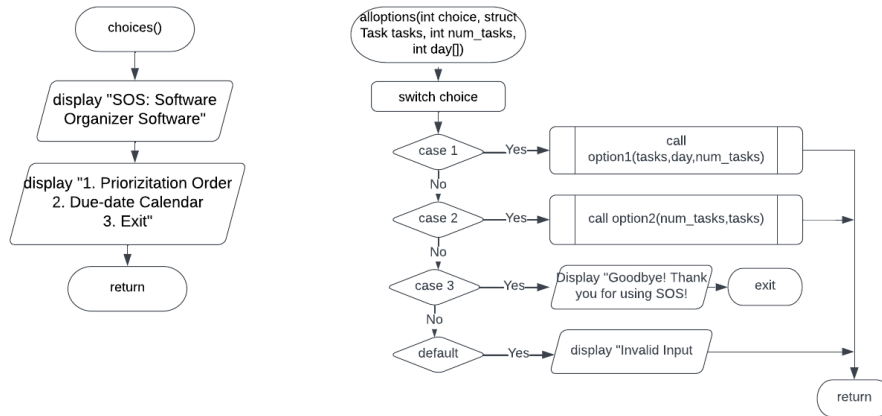


Figure 4

Modules called under alloptions module

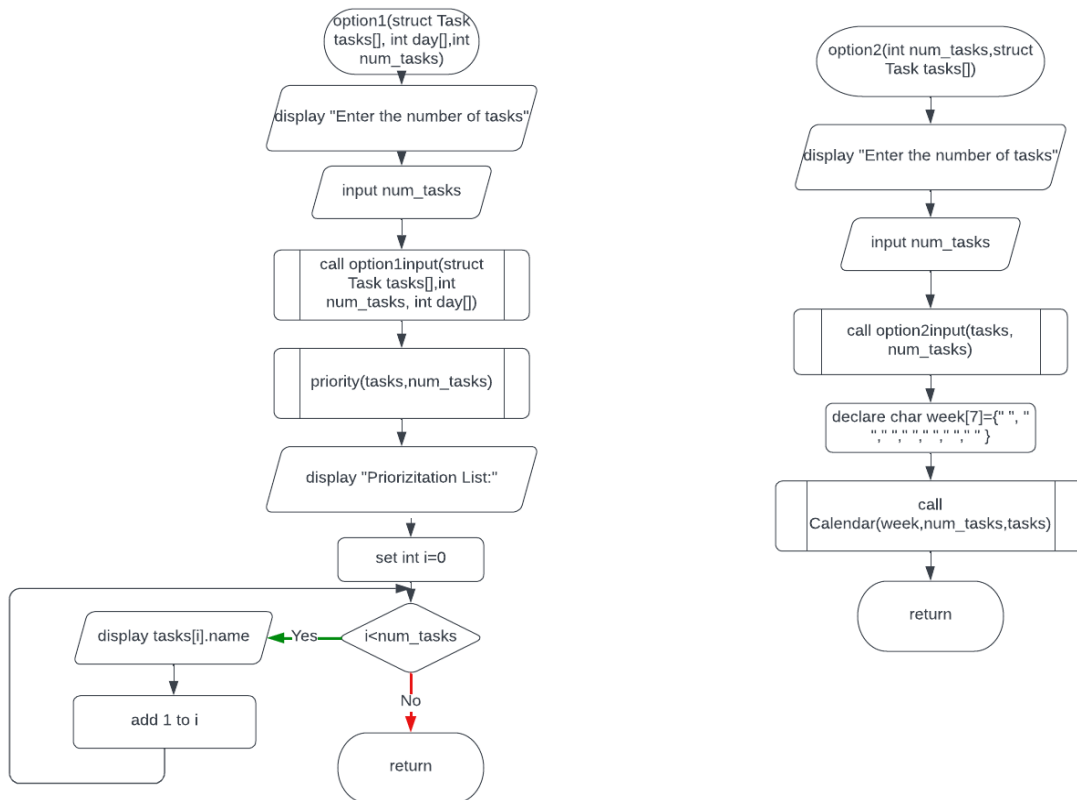


Figure 5

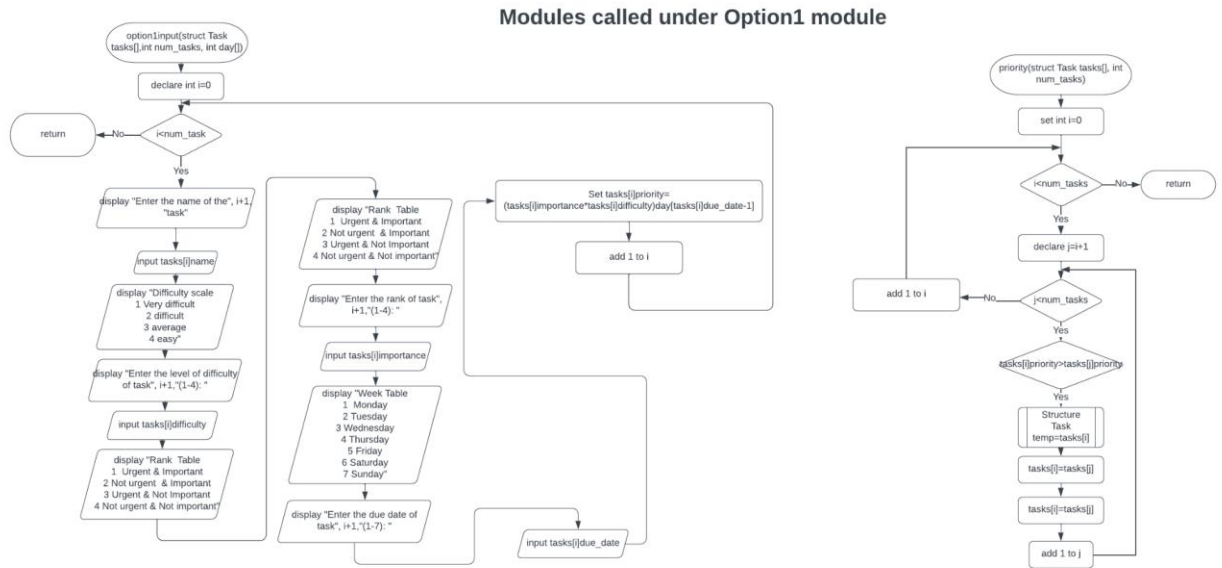
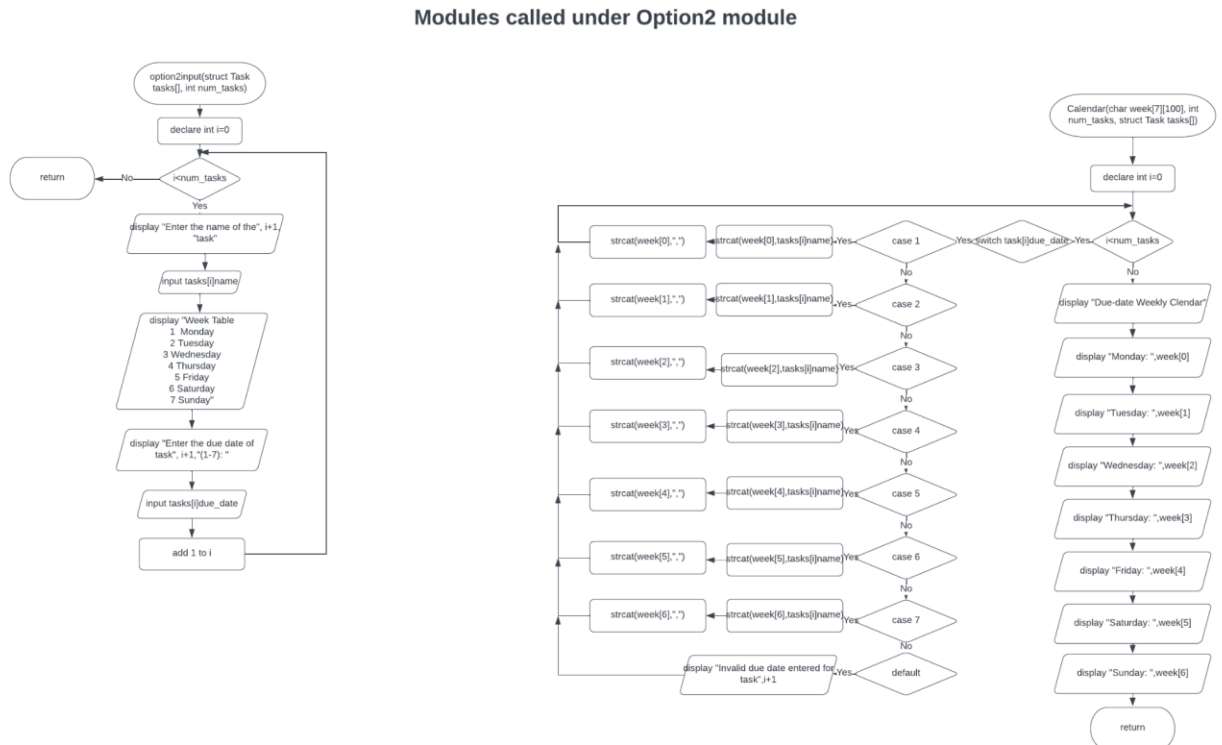


Figure 6



The given code is a C program for a student organizer software. It includes two main options, Prioritization List and Due-date Weekly Calendar, which can be selected by the user. The program uses the following flowchart:

The program includes the standard C libraries and defines a structure, 'Task', which consists of several variables related to a task, including its name, difficulty, importance, due date, and priority. It also defines three functions: 'choices()', 'option1()', and 'option2()'. The 'choices()' function displays a menu for the user with three options: 'Prioritization List', 'Due-date Weekly Calendar', and 'Exit'.

If the user selects the 'Prioritization List' option, the 'option1()' function is called. This function prompts the user to enter the number of tasks and then calls the 'option1input()' function to get information about each task, including its name, difficulty, importance, and due date. After obtaining the information about each task, the 'Priority()' function is called to calculate the priority of each task and sort the tasks in ascending order based on their priority. Finally, the function prints the Prioritization List in ascending order of priority.

If the user selects the 'Due-date Weekly Calendar' option, the 'option2()' function is called. This function prompts the user to enter the number of tasks and then calls the 'option2input()' function to get the name and due date of each task. After obtaining the information about each task, the 'Calendar()' function is called to store the tasks under the appropriate day of the week in the week array. Finally, the function prints the Due-date Weekly Calendar, which displays the tasks due on each day of the week.

If the user selects the 'Exit' option, the program ends. The 'Priority()' function takes an array of Task structures and the number of tasks as input.

It then uses a nested loop to compare the priority of each task with the priority of the other tasks and swaps them if necessary. The 'option1input()' function takes an array of Task structures, the number of tasks, and an array of integers representing the days of the week as input. It then prompts the user to enter the name, difficulty, importance, and due date of each task and calculates the priority of each task using the formula: $\text{priority} = (\text{difficulty} \times \text{importance}) / \text{days to due date}$. The 'option2input()' function takes an array of Task structures and the number of tasks as input. It then

prompts the user to enter the name and due date of each task. The 'Calendar()' function takes an array of strings representing the days of the week, an array of Task structures, and the number of tasks as input. It then stores the tasks under the appropriate day of the week in the week array using a switch statement and prints the Due-date Weekly Calendar.

d. Pseudocode

1 Module Main

2 // Declare variables

3 Declare integer choice, num_tasks, day[7]={7,6,5,4,3,2,1}

4 Declare char pick

5 // Create an array of tasks

6 declare struct Task tasks[MAX_TASKS];

7 // Display the options menu and get the user's choice

8 do {

9 do {

10 call choices()

11 print "Enter your choice (1-3): "

12 read choice;

13// Clear the screen

14 // Execute the selected option

15 call alloptions(int choice, struct Task tasks[],int num_tasks,int day[])

16 }while(choice>3 || choice<1) //to ensure that the input is between 1-3

16 end do-while loop

17 printf("\n\nngo back to menu?(y/n): ")

```
18     read pick

19 } while (pick=='y' || pick=='Y')

20 End Do-while loop

21 printf("Goodbye! Thank you for using SOS!")

22 End Module

23// create a function for the choice of the user

24 module void alloptions(int choice, struct Task tasks[],int num_tasks,int day[])

25 switch(choice) {

26     case 1:

27         call option1(tasks, day, num_tasks)

28         break

29     case 2:

30         call option2(tasks, num_tasks)

31         break

32     case 3:

33

34         break

35     default:

36         print "Invalid choice. Please enter a valid choice (1-3)."

37
```

```
38 end switch-case

39 End Module

40 // Define the task structure

41Module struct Task

42 declare char name[100]

43 declare int difficulty

44 declare int importance

45 declare int due_date

46 declare float priority

47 end module

48// Function to display the options menu

49 module void choices(void)

50 // Print the options available

51 print "SOS: Student Organizer Software"

52 print "1. Prioritization List"

53 print "2. Due-date weekly calendar"

54 print "3. Exit"

55 end module

56

57 // Function to execute option 1
```

```

58 module void option1(struct Task tasks[],int day[],int num_tasks)

59 // Prompt the user to enter the number of tasks

60 print "Enter the number of tasks: "

61 // Read the number of tasks entered by the user

62 read num_tasks

63 // Clear the screen

64 // Prompt the user to enter the details of each task

65 call option1input(tasks,num_tasks,day)

66 // Sort the tasks based on their priority

67 call Priority(tasks,num_tasks)

68 // Print the tasks in ascending order of priority

69 print "Prioritization list:"

70 for i = 0 to num_tasks-1

71     print tasks[i].name

72 end for

73 end module

74 // Function to take the user input for option 1

75 module void option1input(struct Task tasks[],int num_tasks,int day[])

76 for i = 0 to num_tasks-1

77     // Prompt the user to enter the details of each task

```

```
78     print "Enter the name of task ", i+1, ": "

79     // Read the name of the task entered by the user

80     read tasks[i].name

81     // Prompt the user to enter the difficulty level of the task

82     print "Difficulty scale"

83     print "1\tVery difficult\n2\tdifficult\n3\taverage\n4\teasy"

84     print "Enter the difficulty level of task ", i+1, " (1-4): "

85     // Read the difficulty level of the task entered by the user

86     read tasks[i].difficulty

87     // Prompt the user to enter the rank of the task

88     print "Rank table"

89     print "1\tUrgent & Important\n2\tNot urgent & Important\n3\tUrgent & not

90     important\n4\tNot urgent and not important"

91     print "Enter the Rank of task ", i+1, " (1-4): "

92     // Read the rank of the task entered by the user

93     read tasks[i].importance

94     // Prompt the user to enter the due date of the task

95     print "Week table"

96     print

97     "1\tMonday\n2\tTuesday\n3\tWednesday\n4\tThursday\n5\tFriday\n6\tSaturday\n7\tSunday"
```



```

98     print "Enter the due date of task ", i+1, " (1-7): "

99     // Read the due date of the task entered by the user

100    read tasks[i].due_date

101    // Clear the screen

102    // Calculate the priority of the task

103    set tasks[i].priority = (float)(tasks[i].difficulty * tasks[i].importance)
104/day[tasks[i].due_date-1]

105 end for

106 end module

107 // Function to take the user input for option 2

108 module void option2input(struct Task tasks[],int num_tasks)

109 for i = 0 to num_tasks-1

110 // Prompt the user to enter the details of each task

111     print "Enter the name of task ", i+1, ": "

112// Read the name of the task entered by the user

113     read tasks[i].name

114// Prompt the user to enter the due date of the task

115     print "Week table"

116     print
117"1\tMonday\n2\tTuesday\n3\tWednesday\n4\tThursday\n5\tFriday\n6\tSaturday\n7\tSunday"

118// Prompt the user to enter the due date of the task

```

```

119     print "Enter the due date of task ", i+1, " (1-7): ";

120     read tasks[i].due_date

121 // Clear the screen

122 end for

123 end module

124 // Function to execute option 2

125 module void option2(struct Task tasks[], int num_tasks)

126 // Prompt the user to enter the number of tasks

127 print "Enter the number of tasks: "

128 // Read the number of tasks entered by the user

129 read num_tasks

130 // Clear the screen

131 // Prompt the user to enter the details of each task

132 call option2input(tasks, num_tasks);

133 declare week 2-dimensional array char week[7][100]={ "", "", "", "", "", "", "" }

134 call Calendar(week,num_tasks,tasks)

135//function for printing the calendar

135 module Calendar(char week[7][100], int num_tasks, struct Task tasks[])

136 // Loop through each task and append it to the appropriate day of the week

137 For i = 0 to num_tasks - 1

```

```
138     Switch tasks[i].due_date
139         Case 1:
140             strcat(week[0], tasks[i].name)
141             strcat(week[0], ", ")
142         Case 2:
143             strcat(week[1], tasks[i].name)
144             strcat(week[1], ", ")
145         Case 3:
146             strcat(week[2], tasks[i].name)
147             strcat(week[2], ", ")
148         Case 4:
149             strcat(week[3], tasks[i].name)
150             strcat(week[3], ", ")
151         Case 5:
152             strcat(week[4], tasks[i].name)
153             strcat(week[4], ", ")
154         Case 6:
155             strcat(week[5], tasks[i].name)
156             strcat(week[5], ", ")
157         Case 7:
```

```
158         strcat(week[6], tasks[i].name)

159         strcat(week[6], ", ")

160     Default:

161         Print "Invalid due date entered for task " + (i+1) + "!"

162 End Switch

163 End For

164 // Print the weekly calendar with the tasks listed under their respective days

165 Print "\nDue-Date Weekly Calendar"

166 Print "\nMonday: " + week[0]

167 Print "Tuesday: " + week[1]

168 Print "Wednesday: " + week[2]

169 Print "Thursday: " + week[3]

170 Print "Friday: " + week[4]

171 Print "Saturday: " + week[5]

172 Print "Sunday: " + week[6]

173 end module
```

IV. Results

Figure 7

```
SOS: Student Organizer Softwate  
  
1. Priorizitation List  
2. Due-date weekly calendar  
3. Exit  
  
Input the feature you want to see: 1|
```

Figure 8

```
Enter the number of tasks: 2|
```

Figure 9

```
Enter the name of task 2: prologi-project

Difficulty scale
1      Very difficult
2      difficult
3      average
4      easy
Enter the difficulty level of task 2 (1-4): 2

Rank table
1      Urgent & Important
2      Not urgent & Important
3      Urgent & not important
4      Not urgent and not important
Enter the Rank of task 2 (1-4): 1

Week table
1      Monday
2      Tuesday
3      Wednesday
4      Thursday
5      Friday
6      Saturday
7      Sunday
Enter the due date of task 2 (1-7): 1
```

Figure 10

```
Enter the name of task 2: prologi-project

Difficulty scale
1      Very difficult
2      difficult
3      average
4      easy
Enter the difficulty level of task 2 (1-4): 2

Rank table
1      Urgent & Important
2      Not urgent & Important
3      Urgent & not important
4      Not urgent and not important
Enter the Rank of task 2 (1-4): 1

Week table
1      Monday
2      Tuesday
3      Wednesday
4      Thursday
5      Friday
6      Saturday
7      Sunday
Enter the due date of task 2 (1-7): 1
```

Figure 11

```
Priorizitation list:
prologi-project
prologi-exam

go back to menu?(y/n): y|
```

Figure 12

SOS: Student Organizer Software

1. Priorization List
2. Due-date weekly calendar
3. Exit

Input the feature you want to see: 1|

Figure 13

Enter the number of tasks: 5

Figure 14

```
Enter the name of task 1: study-caleng-exam

Difficulty scale
1      Very difficult
2      difficult
3      average
4      easy
Enter the difficulty level of task 1 (1-4): 2

Rank table
1      Urgent & Important
2      Not urgent & Important
3      Urgent & not important
4      Not urgent and not important
Enter the Rank of task 1 (1-4): 1

Week table
1      Monday
2      Tuesday
3      Wednesday
4      Thursday
5      Friday
6      Saturday
7      Sunday
Enter the due date of task 1 (1-7): 2
```


Figure 15

```
Enter the name of task 2: go-to-mall

Difficulty scale
1      Very difficult
2      difficult
3      average
4      easy
Enter the difficulty level of task 2 (1-4): 4

Rank table
1      Urgent & Important
2      Not urgent & Important
3      Urgent & not important
4      Not urgent and not important
Enter the Rank of task 2 (1-4): 4

Week table
1      Monday
2      Tuesday
3      Wednesday
4      Thursday
5      Friday
6      Saturday
7      Sunday
Enter the due date of task 2 (1-7): 4
```

Figure 16

```
Enter the name of task 3: do-prologi-homework

Difficulty scale
1      Very difficult
2      difficult
3      average
4      easy
Enter the difficulty level of task 3 (1-4): 3

Rank table
1      Urgent & Important
2      Not urgent & Important
3      Urgent & not important
4      Not urgent and not important
Enter the Rank of task 3 (1-4): 1

Week table
1      Monday
2      Tuesday
3      Wednesday
4      Thursday
5      Friday
6      Saturday
7      Sunday
Enter the due date of task 3 (1-7): 2
```

Figure 17

```
Enter the name of task 4: pass-project-calculus

Difficulty scale
1      Very difficult
2      difficult
3      average
4      easy
Enter the difficulty level of task 4 (1-4): 1

Rank table
1      Urgent & Important
2      Not urgent & Important
3      Urgent & not important
4      Not urgent and not important
Enter the Rank of task 4 (1-4): 1

Week table
1      Monday
2      Tuesday
3      Wednesday
4      Thursday
5      Friday
6      Saturday
7      Sunday
Enter the due date of task 4 (1-7): 1
```

Figure 18

```
Enter the name of task 5: go-shopping

Difficulty scale
1      Very difficult
2      difficult
3      average
4      easy
Enter the difficulty level of task 5 (1-4): 4

Rank table
1      Urgent & Important
2      Not urgent & Important
3      Urgent & not important
4      Not urgent and not important
Enter the Rank of task 5 (1-4): 4

Week table
1      Monday
2      Tuesday
3      Wednesday
4      Thursday
5      Friday
6      Saturday
7      Sunday
Enter the due date of task 5 (1-7): 3
```

Figure 19

```
Priorization list:  
pass-project-calculus  
study-caleng-exam  
do-prologi-homework  
go-shopping  
go-to-mall
```

```
go back to menu?(y/n): |
```

Figure 20

```
SOS: Student Organizer Softwate
```

1. Priorization List
2. Due-date weekly calendar
3. Exit

```
Input the feature you want to see: 2|
```

Figure 21

```
Enter the number of tasks: 2|
```

Figure 22

```
Enter the name of task 1: prologi-exam

Week table
1      Monday
2      Tuesday
3      Wednesday
4      Thursday
5      Friday
6      Saturday
7      Sunday
Enter the due date of task 1 (1-7): 2
```

Figure 23

```
Enter the name of task 2: prologi-project

Week table
1      Monday
2      Tuesday
3      Wednesday
4      Thursday
5      Friday
6      Saturday
7      Sunday
Enter the due date of task 2 (1-7): 1
```

Figure 24

```
Due-Date Weekly Calendar

Monday: prologi-project,
Tuesday: prologi-exam,
Wednesday:
Thursday:
Friday:
Saturday:
Sunday:

go back to menu?(y/n): |
```

Figure 25

```
SOS: Student Organizer Softwate

1. Priorizitation List
2. Due-date weekly calendar
3. Exit

Input the feature you want to see: 3
```

Figure 26

```
D:\C++\prologi_project pla X + v

Goodbye! Thank you for using SOS!
-----
Process exited after 71.27 seconds with return value 1
Press any key to continue . . . |
```

V. Discussion of Results

The given code is a simple implementation of a student organizer software that offers two options to the user: (1) Prioritization List, and (2) Due-date weekly calendar. The program defines a task structure that includes the name, difficulty level, importance rank, due date, and priority of a task.

The `choices()` function prints out the menu of options to the user and waits for the user's input. The `option1()` function is executed when the user selects the first option, which prompts the user to input task details such as the name, difficulty level, importance rank, and due date. The function then calculates the priority of each task based on their difficulty and importance and sorts them in ascending order of priority using the `Priority()` function. Finally, the sorted tasks are printed out in the console for the user to view.

The `option2()` function is executed when the user selects the second option, which prompts the user to input task details such as the name and due date. The task details are stored in an array called `tasks`. The `Calendar()` function then uses the due date to store the tasks under the appropriate day of the week in the `week` array. Finally, the program prints out the weekly calendar with tasks listed under each day for the user to view.

Overall, the program provides a simple and efficient way to organize tasks based on their priority and due date. The prioritization list allows the user to view which tasks are most important and need to be completed first, while the weekly calendar provides an overview of all tasks due during the week. However, the program can be further improved by adding more features such as reminders or the ability to mark tasks as completed to enhance the overall user experience.

VI. Analysis, Conclusion, and Future Directives

Therefore, efficient time management can help individuals prevent wasting time and lessen psychological pressure brought by deadlines, such as dissatisfaction and anxiety. It can also effectively improve efficiency for students or individuals with high workloads. Additionally, assigning projects based on urgency, importance, as well as the difficulty levels might help students better prioritize their work and develop practical time management skills. Further, students can comprehend the deadline if the due date for input tasks is specified within a week. Moreover, students can finish the tasks on their to-do lists for the following week before it's due. According to Itamar Shatz.(n.d.)Setting a more precise deadline can help individuals stay motivated since realizing how urgent work is can enable individuals to stop putting it off. Additionally, students can effectively prevent forgetting to do specific assignments because they are engaged in other activities by writing down the assignments that need to be finished on the planner. A simple to-do list also enables students to quickly figure out which assignments need to be finished.

For future Directives, it would be more suitable for the code to be able to only ask to input their data before going to the menu. Students may be more proactive in completing tasks if they are aware of more precise deadlines. Additionally, adding a monthly calendar enables students to make long-term plans. Moreover, adding notes to each assignment, enabling them to broadly outline what needs to be accomplished for the task.

References

- A. (2020, August 13). Connection between Self Motivation and Time Management | You Must Know | Digital TK. DIGITAL TK. <https://digitaltk.com/connection-between-self-motivation-and-time-management/>
- Alexander, R, David, F, Ayah, H, Katrin, B.K, (2022, March 15) *Procrastination Among University Students: Differentiating Severe Cases in Need of Support From Less Severe Cases*. <https://www.frontiersin.org/articles/10.3389/fpsyg.2022.783570/full>
- Alyami, A., Abdulwahed, A., Azhar, A., Binsaddik, A., & Bafaraj, S. (2021, March 5). Impact of Time-Management on the Student's Academic Performance: A Cross-Sectional Study. Creative Education; Scientific Research Publishing. <https://doi.org/10.4236/ce.2021.123033>
- Bakioglu, A., & Yildirim, K. (2015). The Relation between Time Management Skills and Academic Achievement of Potential Teachers. Journal of Education and Practice, 6(10), 84-90. Retrieved from https://www.researchgate.net/publication/277741851_The_Relation_between_Time_Management_Skills_and_Academic_Achievement_of_Potential_Teachers
- Bekhet, A. K., & Zauszniewski, J. A. (2012). Mental health of elders in retirement communities: is loneliness a key factor?. Archives of psychiatric nursing, 26(3), 214-224. <https://core.ac.uk/download/pdf/6750696.pdf>
- Correia, M. N. M. M. S. S. a. H. J. M. M. F. S. J. X. R. R. M. D. a. B. (2021, August 24). *The Relationship between Time Management, Work Stress and Work Performance-A Quantitative Study in Portugal*. <https://www.abacademies.org/articles/the-relationship-between-time-management-work-stress-and-work-performance-a-quantitative-study-in-portugal-12182.html>
- Covey, S. R. (2018). The 7 Habits of Highly Effective People: Powerful Lessons in Personal Change. Simon & Schuster. Retrieved from <https://www.simonandschuster.com/books/The-7-Habits-of-Highly-Effective-People/Stephen-R-Covey/9781982137137>

Indeed Editorial Team. (2022, July 22). *What Is Eisenhower's Urgent vs. Important Principle?*

Indeed Career Guide. https://www.indeed.com/career-advice/career-development/eisenhowers-urgent-vs-important-principle?fbclid=IwAR0NcwDy1D_v7A8X22cgqrzFTFsE0XBPP6P_TM3mTKq-JXxdLdpb114FePA

Itamar Shatz.(n.d.). *Deadlines: How Effective Time Constraints Can Boost Productivity.*

<https://effectiviology.com/deadlines/>

McLean: Putting people first in mental, (2022, December 4) *The real reason why you procrastinate*

<https://www.mcleanhospital.org/essential/procrastination#:~:text=The%20issue%20can%20be%20linked,even%20link%20to%20physical%20illness.>

Mind Tools. (n.d.). Eisenhower's Urgent/Important Principle. Retrieved from

<https://www.mindtools.com/all1e0k5/eisenhowers-urgentimportant-principle>

Procrastination Statistics: Interesting and Useful Statistics about Procrastination. (n.d.).

https://solvingprocrastination.com/procrastination-statistics/?fbclid=IwAR3MJMbGdLtCKwUP-uQMzLQQ9W9oiwxl6fRr84MufG_8Scu3mHHXrGEtzlQ

Revolution Prep, (2022, March 23) Priority skills for students

<https://www.revolutionprep.com/blog/prioritization-skills-for-students/#:~:text=Tasks%20that%20are%20urgent%20and,student%20has%20less%20pressing%20responsibilities.>

Sirois, F. M., & Pychyl, T. A. (2013). Procrastination and the priority of short-term mood regulation: Consequences for future self. *Social and Personality Psychology Compass*, 7(2), 115–127. <https://doi.org/10.1111/spc3.12011>

Sleep Foundation. (2023, March 31). *The Link Between Sleep and Job Performance.*

<https://www.sleepfoundation.org/sleep-hygiene/good-sleep-and-job-performance>

Solving Procastiantion, (2023) *Student Procrastination: Why Students Procrastinate and How to Stop It.*

<https://solvingprocrastination.com/student-procrastination/>

Steel, P., & Klingsieck, K. B. (2016). Academic procrastination: Psychological antecedents revisited. *Australian Psychologist*, 51(1), 36–46. <https://doi.org/10.1111/ap.12173>

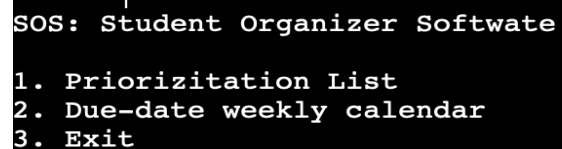
Tice, D. M., & Baumeister, R. F. (2018). Longitudinal study of procrastination, performance, stress, and health: The costs and benefits of dawdling. *Psychological Science*, 29(9), 1608–1618. <https://doi.org/10.1177/0956797618774254>

Appendices

A. User's Manual

When the user enters the SOS, three options are presented. Prioritization List, Due Date Weekly Calendar, and Exit are the available choices in that arrangement.

Figure 27



```
SOS: Student Organizer Software
1. Prioritization List
2. Due-date weekly calendar
3. Exit
```

1. For Prioritization List.

- If user wish to make a Prioritization List, just simply type “1”, Then the program will show the following features:

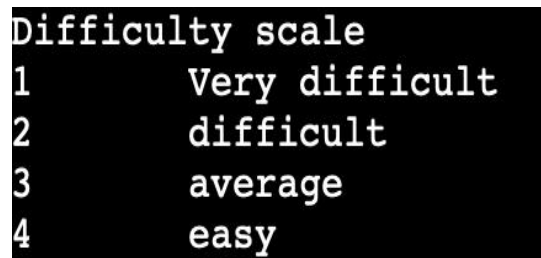
- 1) **Enter the number of tasks** - This feature is for the user to input the number of tasks that he or she wishes to prioritize.
- 2) **Enter the name of the task** - This feature is for the user to input the name for each of the tasks.
- 3) **Enter the difficulty level of task (1-4)** - This feature is for the user to input the difficulty level of the task with 1 for “**Very difficult**” and 4 for “**Easy**”. In addition, the table of the difficulty

will be shown above for the user to know what the levels of difficulty are.

- 4) **Enter the Rank of task (1-4)** - This feature is for the user to input the rank of that particular task with 1 for “Urgent & Important”; 2 for “Not urgent & Important”; 3 for “Urgent & not important”; and 4 for “Urgent & not important” which the table will shown to the user before the user input the rank of the task.
- 5) **Week table** - This feature is for the user to input when is the due date of that particular task that he or she had inputted. 1 for Monday and 7 for Sunday. A week table will be provided for the user.

- If it's just a single task, then it will automatically input the single task as the first task to do.

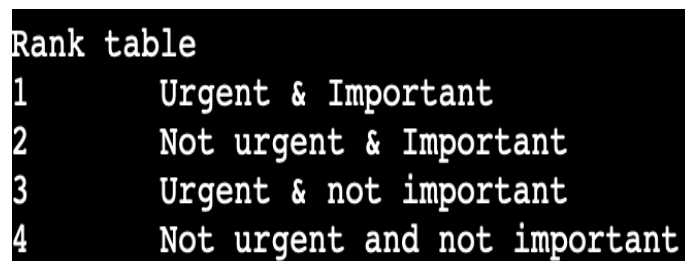
Figure 28



Difficulty scale	
1	Very difficult
2	difficult
3	average
4	easy

(for the difficulty level table)

Figure 29



Rank table	
1	Urgent & Important
2	Not urgent & Important
3	Urgent & not important
4	Not urgent and not important

(for the rank of the particular task)

Figure 30

```
Week table
1      Monday
2      Tuesday
3      Wednesday
4      Thursday
5      Friday
6      Saturday
7      Sunday
Enter the due date of task 1 (1-7): 5
```

(week table for user to input the deadline of the particular task)

- If the user inputted number of task that's more than 1, then after asking the week table of the task 1, the program will then ask the user to input the name, difficulty level, rank, and also the week table for the second task. As follow for the name of task number 3 and so on and so forth.

- For number of task that's more than two :

- 1) After collecting all the needed information, the program will compute it and show which task is the first task that the user needs to work with.

Sample output:

Figure 31

```
Prioritization list:
Art
Math
Science
```

- After showing the Prioritization list, the program will then ask the user if he or she wants to go back to the menu option by simply clicking "y" for going back to the menu option or "n" for exiting the program.

2. For Due-date weekly calendar

- If the user wants to select the Due-date weekly calendar, then the user can just simply type "2" then the program will show the following features:

- 1) **Enter the number of tasks** - This feature asks the user to input the number of tasks that needs to be completed in the week.
 - 2) **Enter the name of the task** - This feature is for the user to input the name for each of the tasks.
 - 3) **Enter the due date of task (1-7)** - This feature is for users to input the due dates of their task that needs to be completed in a week. With **1** for **Monday** and **7** for **Sunday**. A week table will be presented before user input.
- After users input the corresponding task deadlines, the program will be displaying the due date in the weekly calendar. If the number of tasks is more than 1, then the program will be asking the user for the name and due date of the task for the second task and so on and so forth.
 - This is a sample —

Figure 32

```
Due-Date Weekly Calendar  
  
Monday:  
Tuesday: Math,  
Wednesday: Science,  
Thursday:  
Friday:  
Saturday: Litherature,  
Sunday:
```

Figure 33

```
Week table  
1      Monday  
2      Tuesday  
3      Wednesday  
4      Thursday  
5      Friday  
6      Saturday  
7      Sunday
```

this is the week table presented for the user

- After the due-date weekly calendar was presented, the program will then ask the user if he or she would like to go back to the menu, if the users wish to go back ,

then the program will be presenting the 3 options which are the Prioritization List, Due Date Weekly Calendar, and exit the program.

B. Source Code –

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

// Define the task structure
struct Task {
    char name[100];
    int difficulty;
    int importance;
    int due_date;
    float priority;
};

void choices(void)
{
    printf("SOS: Student Organizer Software\n");
    printf("\n1. Priorization List");
    printf("\n2. Due-date weekly calendar");
    printf("\n3. Exit");
}

void option1(struct Task tasks[],int day[],int num_tasks)
{
    printf("\nEnter the number of tasks: ");
    scanf("%d", &num_tasks);
    system("cls");
    option1input(tasks,num_tasks,day);
    // Sort the tasks based on their priority
    Priority(tasks,num_tasks);
    // Print the tasks in ascending order of priority
    printf("\nPriorization list:\n");
    for(int i=0; i<num_tasks; i++) {
        printf("%s\t\n", tasks[i].name);
    }
}
```

```

    }
}
void option1input(struct Task tasks[],int num_tasks,int day[])
{
    for(int i=0; i<num_tasks; i++) {
        printf("\nEnter the name of task %d: ", i+1);
        scanf("%s", tasks[i].name);
        printf("\nDifficulty scale");
        printf("\n1\tVery difficult\n2\tdifficult\n3\taverage\n4\teasy");
        printf("\nEnter the difficulty level of task %d (1-4): ", i+1);
        scanf("%d", &tasks[i].difficulty);
        printf("\nRank table");
        printf("\n1\tUrgent & Important\n2\tNot urgent & Important\n3\tUrgent & not
important\n4\tNot urgent and not important");
        printf("\nEnter the Rank of task %d (1-4): ", i+1);
        scanf("%d", &tasks[i].importance);
        printf("\nWeek table");

        printf("\n1\tMonday\n2\tTuesday\n3\tWednesday\n4\tThursday\n5\tFriday\n6\tSaturday\n7\tSunday"
);
        printf("\nEnter the due date of task %d (1-7): ", i+1);
        scanf("%d", &tasks[i].due_date);
        system("cls");
        // Calculate the priority of each task
        tasks[i].priority = (float)(tasks[i].difficulty * tasks[i].importance) / day[tasks[i].due_date-1];
    }
}

void option2input(struct Task tasks[],int num_tasks)
{
    for(int i=0; i<num_tasks; i++) {
        printf("Enter the name of task %d: ", i+1);
        scanf("%s", tasks[i].name);
        printf("\nWeek table");

        printf("\n1\tMonday\n2\tTuesday\n3\tWednesday\n4\tThursday\n5\tFriday\n6\tSaturday\n7\tSunday"
);
        printf("\nEnter the due date of task %d (1-7): ", i+1);
        scanf("%d", &tasks[i].due_date);
    }
}

```



```

        system("cls");}

    }

void Priority(struct Task tasks[], int num_tasks) {
    for(int i=0; i<num_tasks; i++) {
        for(int j=i+1; j<num_tasks; j++) {
            if(tasks[i].priority > tasks[j].priority) {
                // Swap the tasks
                struct Task temp = tasks[i];
                tasks[i] = tasks[j];
                tasks[j] = temp;
            }
        }
    }
}

void option2(int num_tasks, struct Task tasks[]){
    printf("\nEnter the number of tasks: ");
    scanf("%d", &num_tasks);
    system("cls");
    option2input(tasks,num_tasks);
    // Define an array to store the tasks for each day of the week
    char week[7][100] = {"", "", "", "", "", "", ""};
    Calendar(week,num_tasks,tasks);
}

void Calendar(char week[7][100], int num_tasks,struct Task tasks[])
{
    // Store the tasks under the appropriate day of the week in the week array
    for(int i=0; i<num_tasks; i++) {
        switch(tasks[i].due_date) {
        case 1:
            strcat(week[0], tasks[i].name);
            strcat(week[0], " ");
            break;
        case 2:
            strcat(week[1], tasks[i].name);
            strcat(week[1], " ");

```

```

        break;
    case 3:
        strcat(week[2], tasks[i].name);
        strcat(week[2], ", ");
        break;
    case 4:
        strcat(week[3], tasks[i].name);
        strcat(week[3], ", ");
        break;
    case 5:
        strcat(week[4], tasks[i].name);
        strcat(week[4], ", ");
        break;
    case 6:
        strcat(week[5], tasks[i].name);
        strcat(week[5], ", ");
        break;
    case 7:
        strcat(week[6], tasks[i].name);
        strcat(week[6], ", ");
        break;
    default:
        printf("Invalid due date entered for task %d!\n", i+1);
        break;
    }
}

printf("\nDue-Date Weekly Calendar\n");
printf("\nMonday: %s\n", week[0]);
printf("Tuesday: %s\n", week[1]);
printf("Wednesday: %s\n", week[2]);
printf("Thursday: %s\n", week[3]);
printf("Friday: %s\n", week[4]);
printf("Saturday: %s\n", week[5]);
printf("Sunday: %s\n", week[6]);
}

void alloptions(int choice, struct Task tasks[],int num_tasks,int day[]){
    switch(choice){
    case 1:

```

```

        option1(tasks,day,num_tasks);

        break;
    case 2:
        option2(num_tasks,tasks);
        break;
    case 3:
        printf("Goodbye! Thank you for using SOS!");
        exit(1);
        break;
    default:
        printf("invalid input");
        system("cls");
        break;
    }
}
int main() {
    int num_tasks = 0,choice;
    char pick;
    int day[7]={ 7,6,5,4,3,2,1 };
    do{
        do{
            choices();
            printf("\n\nInput the feature you want to see: ");
            scanf("%d",&choice);
            system("cls");
            struct Task tasks[num_tasks];
            alloptions(choice,tasks,num_tasks,day);
        } while (choice>3 || choice<1);
        printf("\n\nngo back to menu?(y/n): ");
        scanf("%s",&pick);
    } while (pick=='y'||pick=='Y');
    printf("Goodbye! Thank you for using SOS!");
    return 0;
}

```


C. Work breakdown

Table 2

Student Name	Tasks Assigned	Percentage of the Work Contribution
Barja, Samuelle P.	<p>Introduction-all</p> <p>Review of Related Literature- Eisenhower's Urgent/Important principle, Negative Effects of Procrastination</p> <p>Methodology- Hierarchy Chart, Flowchart,Pseudocode</p> <p>Source Code</p> <p>Results & Discussions</p>	100%

Chua, Melody	Hazel	Review of related Literature- Significance of Time Management Methodology-IPO Table Analysis, Conclusion, & Future Derivatives User's Manual Code-Helped in organizing the modules	100%

Uy, Kae Anastasha T.	Review of Related Literature Flowchart-Helped in organizing	90%
-------------------------	--	-----

D. Personal Data Sheet



Name: Samuelle P. Barja

Email Address: [samuelle_barja@dlsu.edu.ph](mailto:samuella_barja@dlsu.edu.ph)

Course: Computer Engineering



Name: Hazel Melody Chua

Email Address: hazel_melody_chua@dlsu.edu.ph

Course: Computer Engineering



Name: Kae Anastasha T. Uy

Email Address: kae_uy@dlsu.edu.ph

Course: Computer Engineering

RUBRIC FOR TERM PROJECT

CRITERIA	EXEMPLARY (90-100)	SATISFACTORY (80-89)	DEVELOPING (70-79)	BEGINNING (below 70)	WEIGHT
Experimental Plan (Flowchart/ Algorithm) <i>(SO-PI: B1)</i>	Experimental plan has supporting details and diagram/algorithm that is stated and well explained	Experimental plan has supporting details and diagram/algorithm that is stated but not explained	Experimental plan is vague or brief. It has supporting details and does not have diagram/algorithm	No experimental plan presented	20%
Codes/Data/Program	The program utilized some appropriate models or equations to in solving problems. Data is well utilized in the program. Program code are easy to read. Program output has no error. Questions are answered completely and correctly	The program barely utilized some equations in solving problems. Data is somewhat utilized in the program. Program code are easy to read. Program output has an output but logically incorrect. Some questions are answered completely and correctly	Data is not utilized in the program. It has a missing significant code/syntax in the program.	No program presented	30%
Use of Appropriate Tools and Techniques <i>(SO-PI: K1)</i>	Appropriate tools and techniques are properly used for all aspects of the project	Appropriate tools and techniques are used in most of the aspects of the project and all of these are used	Appropriate tools and techniques are used in majority of the aspects of the project	Appropriate tools and techniques are used in less than half of the aspects of the	10%

		properly	and all of these are used properly	project and/or tools are not used properly in at least half the aspects of the project	
Project Documentation	Project documentation is orderly presented starting from statement of the problem, to objective of the project, followed by review of literature, design consideration, presentation of data or output and conclusion. The report was grammatically correct, logically presented and used the required format.	Project documentation is complete with statement of the problem, objectives, design consideration, presentation of data and output and conclusion. The report had minimal grammatical errors and somewhat presented logically. The required format was used.	Project documentation is basically limited to algorithm presentation of data and output but no basis of the design was presented. The report had a lot of grammatical errors and not logically presented; the required format was barely used.	Project documentation is not reflective of algorithm design and/or characterization. The report had a lot of grammatical errors, was not logically presented and the required format was not used.	30 %
Project Presentation	Project presentation is complete and backed up by complete design consideration, logic formulation and review of related literature	Project presentation is complete with algorithm simulation results backed up by design considerations.	Project presentation shows a system completely simulated but is not backed up by clear explanation of how algorithm was derived	Project presentation lacks clarity in terms of presenting and characterizing the behavior of the algorithm	10 %
TOTAL					100%