

**(TEB 1113 / TDB 1023) - ALGORITHM AND DATA STRUCTURES**

**JANUARY 2021 SEMESTER**

**PROJECT – FINAL REPORT**

|  |  |  |
| --- | --- | --- |
| **NAME** | **STUDENT ID NUMBER** | **PROGRAMME** |
| Tan Hon Chi | 19000788 | Computer Engineering |
| Tan Yu Ting | 19000505 | Computer Engineering |
| Rayan Roy Edward | 19000652 | Computer Engineering |

**TABLE CONTENT**

|  |  |  |
| --- | --- | --- |
| NO. | DESCRIPTION | PAGE |
| 1 | Introduction | 3 – 4 |
| 2 | Coding Discussion | 5 – 32 |
| 3 | Screenshot of Output | 33 – 38 |
| 4 | Timeline Work | 39 – 44 |

1. **INTRODUCTION**

In this Algorithm and Data Structures Project, we are required to create and complete an Event Participation text – based application. By completing this application, Online GDB web will be used to compile coding and get the correct output. Besides, Jupyter Notebook will also be used to create graph coding and bar graphs animation result which are created by Bubble Sort and Quick Sort algorithms.

Before starting to compile the coding, we have started our first discussion and done the decision on doing an event which is named Run for Nature. This event is not only a race competition which has two categories racing competition, but also is organized to invoke people to take the responsibilities for protecting the environment. Before starting the marathon competition, participants will have to fill up the registration form which is used to collect the participants’ information and this information will be arranged and done by storing them into files so that the output of the coding will be clearer and better. First, participants will have to insert their name, age, ID number, email address and contact number which are used for organizers references. This information will be stored into a file named “Biodata File.txt” which will display the biodata table output.

Besides, our coding will also provide the option on deleting the specific information that is inserted wrongly by the participant. For example, participants inserted wrongly their name, they are allowed to correct it by choosing option 5 which is able to change and correct the wrong information. Other than that, we have also provided the options on deleting the information that is inserted wrongly by the participant. Those participants are able to delete the information by inputting their position or ID number, then the person’s information will be deleted and they are required to insert their complete information one more time.

Furthermore, we have also included the options which are used to display and explain our event information. Information on organizers’ information and event description will be displayed to participants so that participants will be clearer on our event purpose and event information. Apart from that, we have created a feedback form which is used to collect the satisfaction of our registration form from participants. They are required to choose the satisfaction level of our registration form and give us their suggestion on improving our registration form. These feedbacks will also be stored in a file which is named as “Feedback File.txt” so that the output of the feedback information will not be messy and we are able to read all the feedback properly.

Other than that, after completing the marathon competition, participants are required to insert their time taken for the marathon. Before inserting the time taken result, participants will have to insert their ID number first because this insertion will be used for displaying the participant’s name when the output is completed. After inserting the ID number, time taken input will have to be completed by the participants. Next, when the program is end and all the result have been inserted, time taken result for the marathon will be displayed in a file which is named as “Time\_Taken.txt”. The name and time taken result will be showed in a table which is fixed by the setw() function which has the function of defining the number of characters position so that the table format will not be affected by different length of name and time taken.

Besides, we have also included the algorithm and calculation concepts which are used to sort the data, form the interested animation graphs and complete the calculation that are done by the Queue Data Structure. The algorithms that are done in our coding are Quick Sort Algorithm and Bubble Sort Algorithm which are used to sort the id data number and create two animation bar graphs that represent the sorting process. Other than that, Binary Search Tree are also used to sort and search the available ID number. Binary Search Tree is an algorithm that form a tree graph which has the concept on if a node number is smaller than the root number, the number will be placed as the left subtree of the root, else the node number will be the right subtree of the root.

1. **CODING DISCUSSION**

**GDB Coding Link** : https://onlinegdb.com/HyehTTJWBd

In this project, we have completed 6 .h files which are Event.h, All\_string\_related.h, Queue.h, linkedList.h, BST.h and Graph.h. Other than that, we have also created 6 text files which are used to display the output result and these files are declared by the fstream data type that is able to create files, write information into the files and read information from the files. The .txt files that have been created are Lucky Number File.txt, Time\_Taken.txt, Biodata File.txt, Bubblesort\_data.txt, Quicksort\_data.txt and Feedback File.txt.

Through this project, we have learnt more knowledge about the differences of the linked list and the array. For the difference on execution time, we know that if certain elements have been declared and this condition is being modified in arrays, the execution time will be faster and better because it is able to direct access the declared size, while for linked list, it will take some times because all the previous elements will have to be traversed to reach the element. Therefore, insertion and deletion operations are suitable to be done in linked list because the size will be increased or decreased by depending the number of insertion and deletion.

**2.1. Event.h**

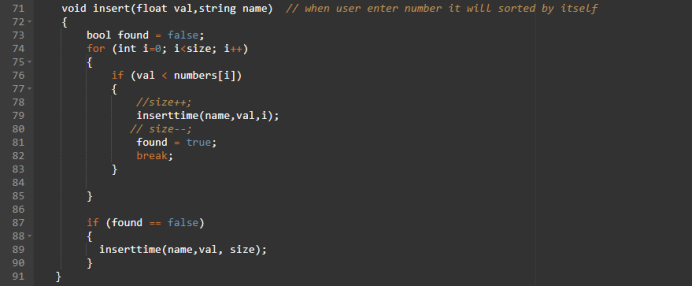
A general view on Event.h is basically using array and normal algorithm. In this Event.h file contains 9 functions algorithm, 5 arrays and size where it will be used as global variable in the Event.h file. Besides that, we declare all the function as public thus it can be access from anywhere. Basically this event will get all the data from the linked list and store in the array in the Event.h .

**2.1.1. void lucky\_number(string \*nameS, int size2)**



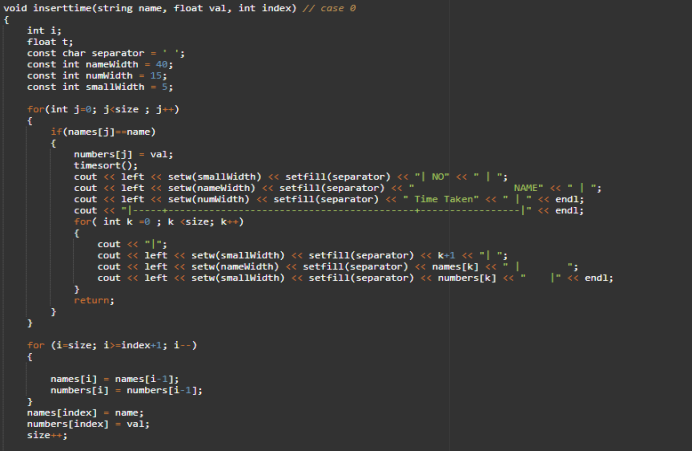
This is void luck number the number of participants that join the event. Basically, the summary of the function is that we want to pre assigned their lucky number by using rand function where it will generate random number and it will show in the output and store in the file which call Lucky Number File.txt. In line 29 to line 42 we use do -while loop because to make sure the that lucky number is not been repeated. After that, the lucky number will store in a array. Thus, the complexity of this function is O(N^2) where we are using 2 for loop.

**2.1.2. void insert (float val, string name)**



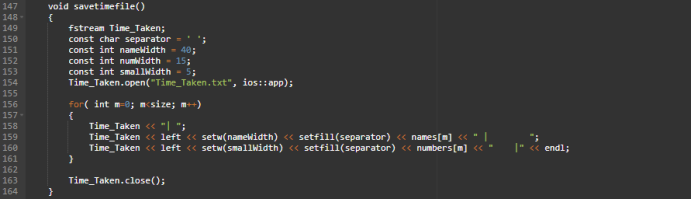
The purpose of this function is when user will enter their taken time to finish their marathon. beside that it will find the sorted according to the time taken in ascending order follow by their index number. This function will find the index number which smaller and call inserttime () which passing 3 arguments which is (name of the participants, time taken, index) = (name,val,i). The complexity of this function is O(N) where we are using one for loop.

**2.1.3. inserttime(string name, float val, int index)**



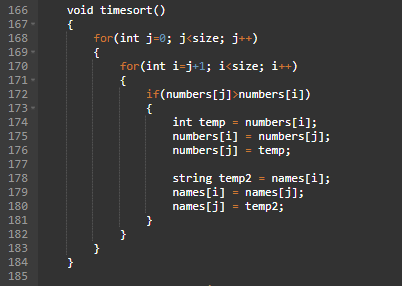
The purpose of this function is it will insert the name and the time according to the sorted way where this function will get the information from the void insert (float val,string name). So, after getting the information for the insert () it will insert according to the index. We include the timesort() where it will swapping process with the value and names. The complexity of the function is O(N).

**2.1.4. Void savetimefile()**



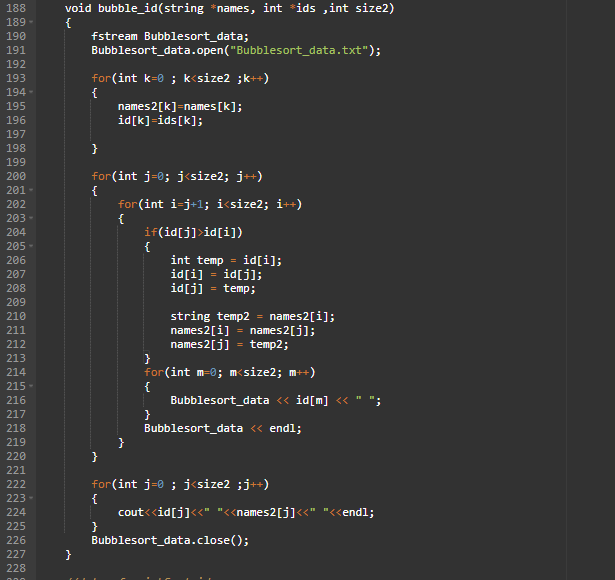
This function will the save the output in a file which call as Time\_Taken.txt. Setw () and setfill() is to adjust the size of the table. Basically, to use this function we need to use #include <iomanip> library to print the output and save in file. The complexity of this algorithm is O(N).

**2.1.5. void timesort()**



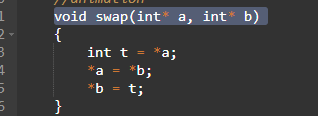
The Function of timesort is to do sorting and swapping the name and the time taken. The complexity of this function is O(N^2).

**2.1.6. void bubble\_id(string \*names, int \*ids ,int size2)**



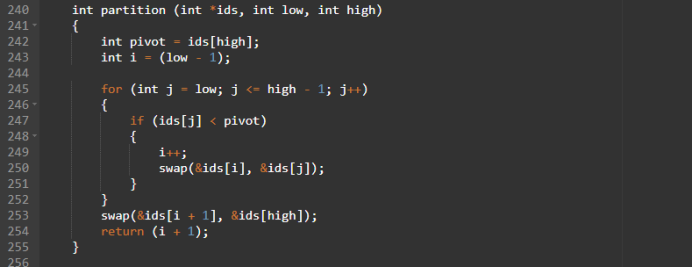
The purpose of this function is to do the sorting id number of the participants and the same time when want to verify how long is the process of the sorting is been occur for a bubble\_sort algorithm. Thus, it the process of sorting will be save in the Bubblesort\_data.txt. The complexity of this algorithm is O(N^2)

**2.1.7. void swap(int\* a, int\* b)**



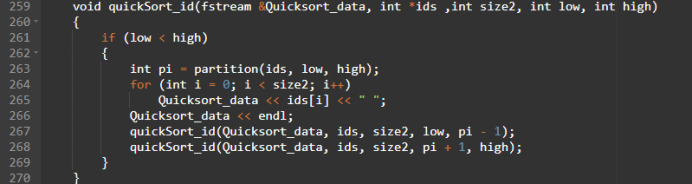
This function is about swapping between numbers where int t is will temporarily will store the value. the complexity of this is O (1).

**2.1.8. int partition (int \*ids, int low, int high)**



This function uses the last element as the pivot, positions it in the proper position in the sorted list, and moves all smaller (smaller than pivot) elements to the left of pivot and all larger elements to the right of pivot. The complexity of this is O(N) because we are using for loop where it will run n times. Then it will return the value to the function that call it.

**2.1.9. void quickSort\_id(fstream &Quicksort\_data, int \*ids ,int size2, int low, int high)**

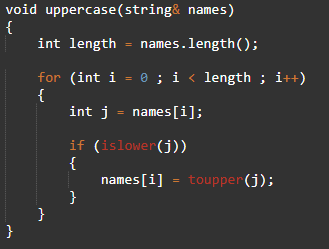


This function needs 4 parameter which is fstream &Quicksort where it will store all the sorting process in the file. We take information \*ids from the linked list and do a for loop which we will base on the size and declare 2 integer variable where is low and high. Basically, quickSort is a partition array and it will recursively call itself twice to sort the two resulting subarrays. This algorithm is quite efficient for large-sized data sets as its average and complexity are O(N^2), respectively. Because we call the same function for 2 times.

**2.2. All\_string\_related.h**

All\_string\_related.h is a file that regarding to event information and details that need to be known by the participants. The functions that are created for event details in this file are organizerInfo() and Frequently\_asked\_question(). Besides, feedback() has also been created in this file which is used to collect the feedback and get the suggestions from participants. For the feedback(), we have included the while loop which is used to display the “Invalid Input!” for the inputs that are not in the options of the satisfaction part.

Other than that, uppercase() function is also included and done in All\_string\_related file which is mainly applied to change the string letters from lowercase letters to uppercase letters. The process of completing the letter change is done by checking whether the string contains lowercase letters. If there are any lowercase letters, these letters will become uppercase letters, and if the uppercase letters are completed, these letters will be maintained and kept.

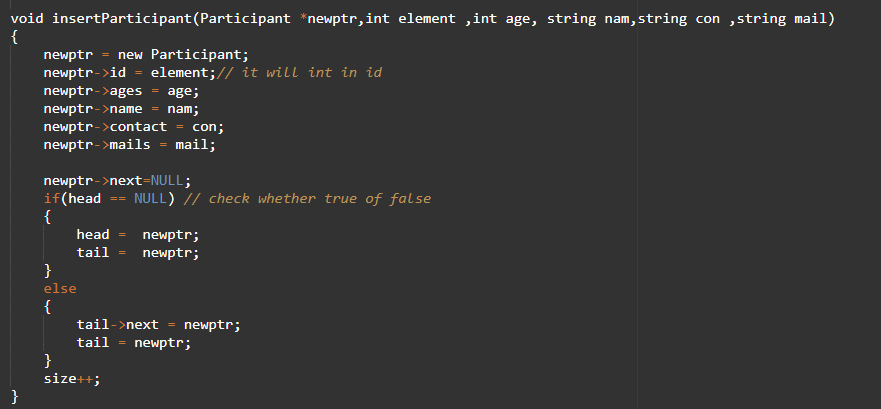


Based on this coding, islower() is used for recognizing the lowercase letters while toupper() is used for changing and converting the lowercase letters to uppercase letters.

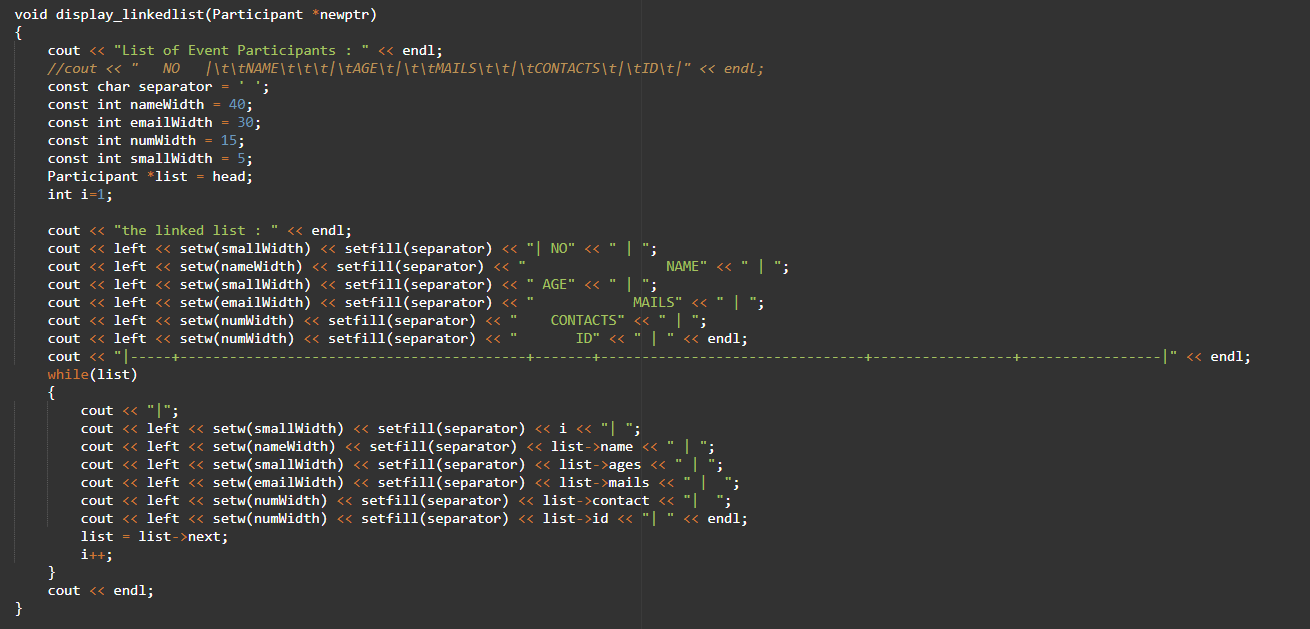
**2.3. linkedList.h**

In this file, the methods are all designed for linked list. The pointer in linked list will be representing the participants where the class attributes are the biodata of the participants. The example of attributes are integer variables of id and ages, string variables of name, contact and mails and the pointer variable named next. In the class several variables are declared such as the id and ages as integer list, size as integer variables and initialize zero and names as string list. The two pointers of type Participant named head and tail are also declared.

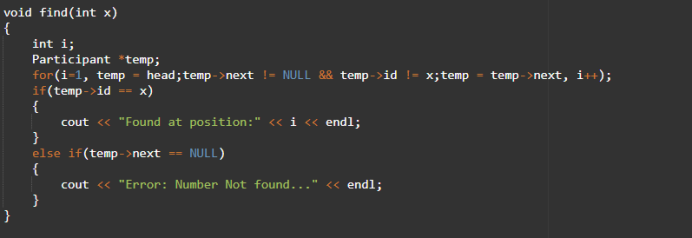
**2.3.1. void insertParticipant(Participant \*newptr,int element, int age, string nam,string con, string mail)**

This function is to allow the user to input their biodata and save the data into linked list. Thus, the parameters of this function are the pointer named newptr and the biodata of participants such as element(id), age(ages), nam(name), con(contact) and mail(email). Assign all the value into attributes of newptr and connect newptr into the linked list according to certain conditions. Variable size will have increment of 1.

**2.3.2. void display\_linkedlist(Participant \*newptr)**

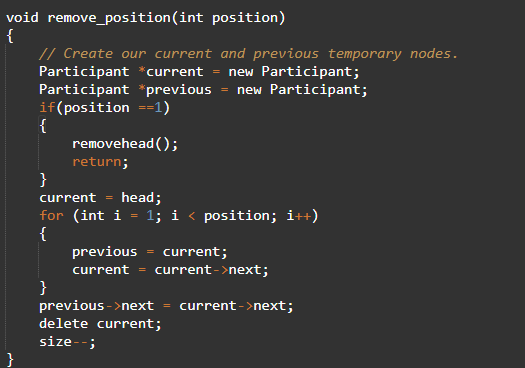
This function will display all the biodata input by the participants. In order to print the output in an orderly manner, the setw() and setfill functions from the iomanip library are used so that the output can be displayed nicer. Besides that, we also have a function void savefile\_linkedlist() where this function will save all the biodata input by users into a file named biodata\_file using fstream library.

**2.3.3. void find (int x)**



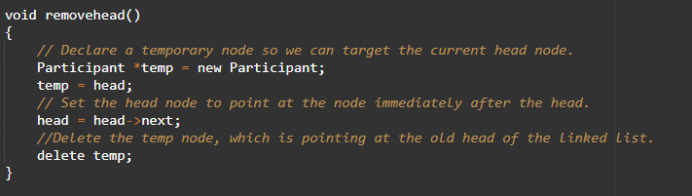
This finds function basically to find the input id of the participant at which index. So basically, the worst-case complexity will be O(N) because it will need to run n times to find the id participant which have be store the memory address.

**2.3.4. void remove\_position(int position)**



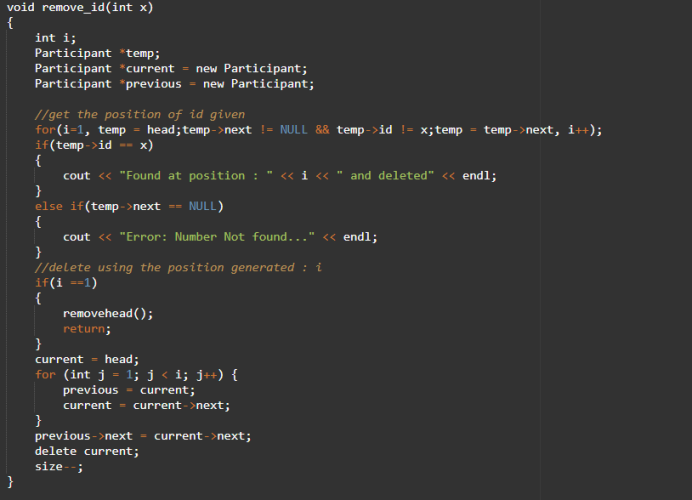
The purpose of void remove\_position(int position) is removing all the data for the specific participant which will refer to position of the name. If the position at number 1 then calls the removehead () else use for loop to find the position, then delete the current node using delete function and connect with the previous node. The complexity for this algorithm will be O(N) because using a for loop.

**2.3.5. void removehead()**



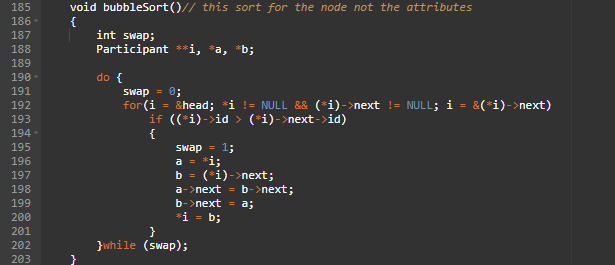
This will remove the node which will be the first position and the complexity of this is O (1).

**2.3.6. void remove\_id(int x)**



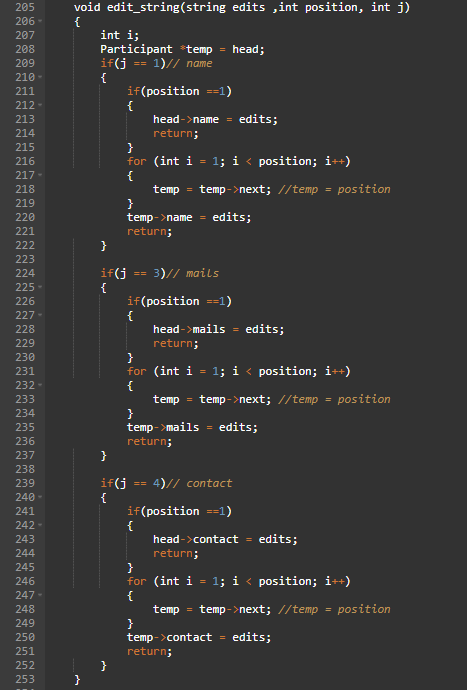
This is a combination of removehead (), find () and delete (). Basically, user want to delete a data. The user just needs to key their id number and this algorithm will the id whether same. If same then it will delete the node that contain the specific participant bio\_data. Something it will show number not found means that user enter wrong id or else the bio data does not exit. The complexity for this algorithm is O(N) where it run for n times until it finds the id or the specific participants.

**2.3.7. void bubbleSort ()**



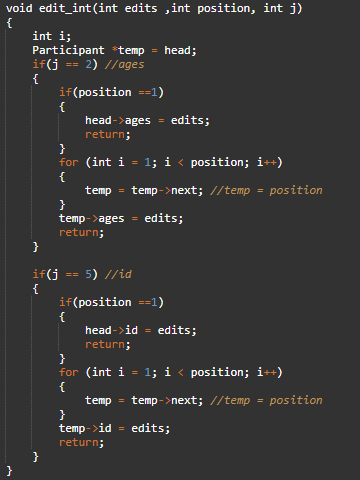
This bubbleSort is just same where it will do the sorting of the data. But this is a bit different because we are using to compare the id only. So, the id of participants is smaller than it needs to be change follow the rest of the bio\_data. So basically, we come up with idea by comparing the id the swap or sort by using node. So that is why we did 3 pointers in this function. where I will represent the flow of the data and can recall or go ahead of the data. Pointer a and b is is where it will change the node which contain the bio\_data for each participant. Basically, the worst-case complexity is O(N^2).

**2.3.8. void edit\_string(string edits, int position, int j)**



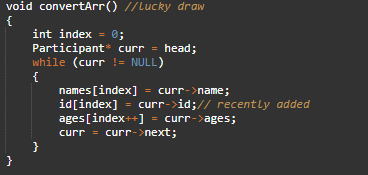
This purpose of this function is to edit or update the biodata for specific participant. For example, it in this function has 3 parameters where the first on is string edits where the user will let us know the string that need to update, then second parameters is int position where the index of the participants. Lastly int j where the option of the user that need to update or edit. For example, if the user wants to update their name just press one where it will go the name function. So basically, the complexity for this algorithm is O(N).

**2.3.9. void edit\_int(int edits, int position, int j)**



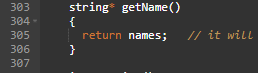
This function is same with the void edit\_string() but only different is that this function will edit for int.

**2.3.10. void convertArr()**



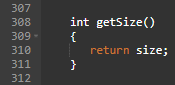
This convertArr() algorithm is about how to convert data linked list into an array because by changing it into array it will be helpful for BST.H and Event.h where there needs the access of the information of the participants biodata. The curr will access the attributes to get data and store them into the array. The worst complexity is for this function is O(N) where N refers to the curr where it will run for many times until it reaches to null then it will terminate the loop.

**2.3.11. string\* getName()**



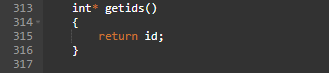
This function will return array of name that contain in obj1

**2.3.12. int getSize()**



This function will return size of the array.

**2.3.13. int\* getids()**



This function will return array of ids that contain in obj1

**2.3.14. int\* getages()**

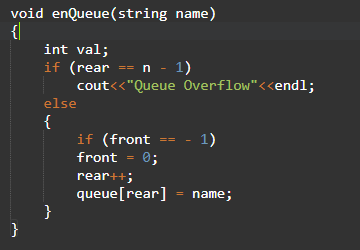


This function will return array of ages that contain in obj1

**2.4. Queue.h**

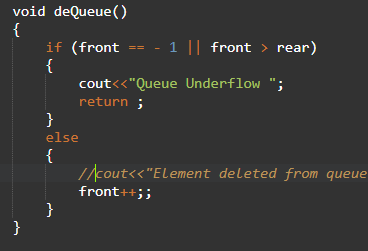
Queue.h is a file with functions named enQueue(), deQueue() and displayQueue(). These 3 functions are designed according to the characteristic of the special list, queue where it pushes an element into the back of the queue and removes an element from the front of the queue. In the class named Queue, declare the string list named queue[1000] and integer variable of n, front and rear.

**2.4.1. void enQueue(string name)**



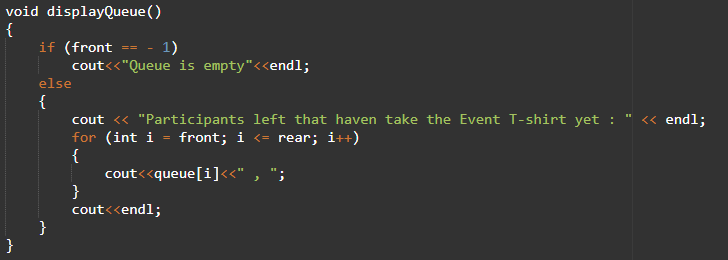
The paramter is a string variable, name and the enQueue function is used to input the variable into the list named queue from the back. The statement queue[rear] = name will input the string varible into the queue according to the value of integer variable rear.

**2.4.2. void deQueue()**



The deQueue function will remove the element from the list named queue from the front of the queue. The front++ will increase the value of integer variable front by 1 and the element with index less than front will be ignored.

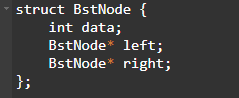
**2.4.3. void displayQueue()**

The displayQueue() function will print the list named queue according to the value of integer variables front and rear where the elements in queue with index starting from front to rear will be displayed.

**2.5. BST**

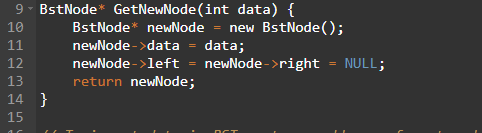
In general overview of binary search tree or BST.h file is about to insert the id of the participants and the search the whether the id is exit or not. Basicalle. in this file we have 1 class where it has all the attributes such as the left,right and etc. Then in this file we have 3 function which is GetNewNode( ), Insert(), Search() .

**2.5.1. struct BstNode**



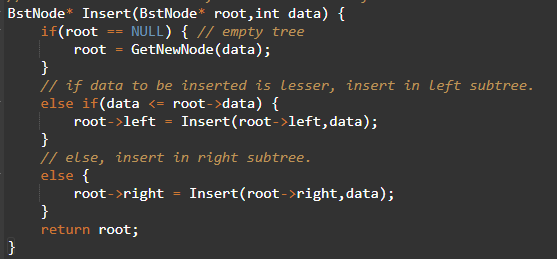
Basically, this are the attributes that we have been used for this coding basically we create because we will be using this type of symbol “->”. where the function that is to store and find the value.

**2.5.2. BstNode\* GetNewNode(int data)**



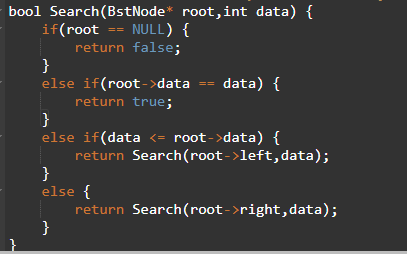
Basically, if the initially the node is null or empty then it has to create a new node. newNode pointer have been created and it will store the data from the user and then it will create the newNode for left and right and assign it as NULL. Finally, it will return the newNode to the function that call this. The complexity of this is O (1).

**2.5.3. BstNode\* Insert(BstNode\* root,int data)**



Basically, of getting the data from the user, we need to 3 type of conditional statement If the tree is empty then we need to call GetNewNode(). The second statement is the data is smaller than the root-> data, where it will be store in the left child. Finally, if the root-> data is smaller den it will be the right child. Basically, to insert in left or right child we will do recursive to insert the data. So basically, the worst-case complexity will be O(N) where N is the one where it will call its own function.

**2.5.4. bool Search(BstNode\* root,int data)**

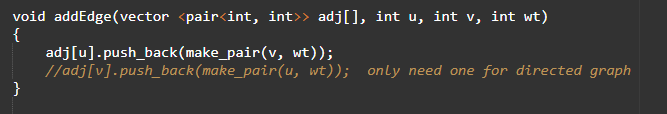


Summary of this function we did a boolean function where it will return true or false when find the data/ id of the participants. We will check whether the data is smaller than the root then it need to search at the left child else it must be search at the right chill. So basically, this function complexity is O(N) because we are using recursive method where it will call its own function.

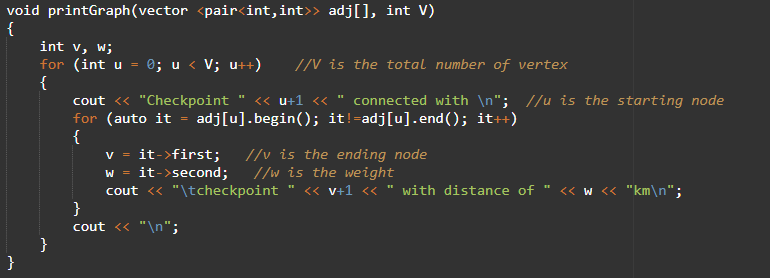
**2.6. Graph.h**

Graph.h is a file with functions named addEdge() and printGraph(). This is because a graph is constructed from nodes and edges. In this class, we will produce a graph which represents the running route for the event. In order to design the graph, vector library is included.

**2.6.1. void addEdge(vector <pair<int, int>> adj[], int u, int v, int wt)**

This function carries 4 parameters which is the vector named adj[] and the integer variables of u, v and wt. Variables u and v represent the nodes where the wt represents the weight or distance of the edge between 2 nodes. In order to append the edges, push\_back and make\_pair functions are applied to design the graph structure. The push\_back() function is used to push elements into a vector from the back while the make\_pair function returns a single object that contains the two items.

**2.6.2. void printGraph(vector <pair<int,int>> adj[], int V)**

This function is to display the relationship between the nodes regarding the connectivity and the weight. In the function, variable u represents the starting node, v represents the ending node and w represent the weight between u and v. The vector adj[] and integer variable V is passed as parameters where V is the total number of vertices. In the pair, the first element is referenced as ‘first’ and the second element as ‘second’ in fixed order as (first, second). This, the first and second in the code refer to the element in the pair.

**2.7. Lucky Number File.txt**

Lucky Number File.txt is created for displaying the lucky numbers which are assumed randomly by the rand() function. The function that is used to call the text file is lucky\_number() which is created in the Event.h file. When the lucky\_number\_file is completely declared and opened, the naming of the file will be created and the ios::app which is used for appending the new data to the current data of the file will also be applied. Therefore, the data of the list will not be deleted and we are able to have the complete lucky numbers data. Participants who have completed the insertion process will be assigned lucky numbers randomly and the list of lucky numbers that contains name of the participants and lucky numbers will be stored and displayed in the text file.

**2.8. Time\_Taken.txt**

Time\_Taken.txt will be used to store the results of participants which the participants will insert their time taken result after they have completed the marathon. By having this text file, savetimefile() has been created and the fstream data type is declared inside the function. After opening the file, the data of the time taken result will be stored into Time\_Taken.txt and the list will be formed by following the length of width that are declared by us. Besides, setw() and setfill() are used to set the table list so that the table list will not be affected by different length of name and time taken result. Other than that, application of ios::app which is used to append the new data to the old data of the file is very important so that when the insertion is done in different round of times, the current data fill will not be deleted. Therefore, storing function of the file for a complete data list is succeeded. Lastly, the file will be closed by the close() when the program is completely end.

**2.9. Biodata File.txt**

Biodata File.txt will be the file that is used for storing and displaying the data list of participants’ information. This file is created and called by the savefile\_linkedlist() function which is done in the linkedList.h file. The file will be declared by the fstream data type which is included the functions of input, writing the information into the file and output, read the information from the file and display the biodata list. Besides, the insertion operation will be done by the participants in different number of times, so the file will keep having new biodata input. In this case, to avoid the current data is not deleted by the new insertion, ios::app will have to be applied in the file open process. ios::app is used to append the new data into the file without deleting the previous data.

**2.10. Bubblesort\_data.txt**

Bubblesort\_data.txt stores the process of sorting id numbers which we inserted 10 id numbers for sorting and this file is created and called by the bubble\_id(). Through the sorting process that is done by bubble sort algorithm, that is able to know that the sorting process is quite long and it takes more time to complete the sorting. For this file, we do not apply the ios::app mode which is used for appending the output in the file. This is because the bubble sort will just be run for one time to get the complete data and sorting process for the 10 id numbers.

**2.11. Quicksort\_data.txt**

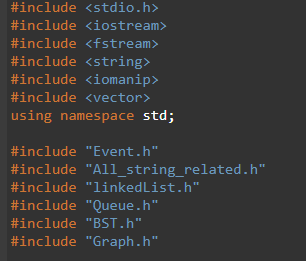
Quicksort\_data.txt file will show the process of sorting for 10 id numbers that are inserted. Besides, this file is called by the main() which the case 15 has been included by the declaration of file, file naming and open(). Based on the data file, it shows that the process of quick sorting of the 10 id numbers is faster than bubble sorting and it does not have to spend more time on sorting the 10 id numbers. This is because it is more efficient on dealing the huge lists and the it will not require any additional storage for sorting process. Quicksort\_data.txt file is same as the Bubblesort\_data file which it is also not applied the ios::app mode. This is because the quick sort will only be inputted and run for one time. Therefore, there will not have the operation of new insertion and result of previous data.

**2.12. Feedback File.txt**

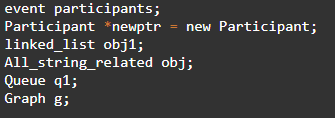
Feedback File.txt is used to store and display the outputs feedback that are inserted and done by participants. In this feedback file, it includes the information of participants’ name, satisfaction of the registration form and the suggestions of improving the registration form. This file is opened and named by the feedback() that includes the declaration of the fstream and open(). Besides, the ios::app mode is also used for this file because the insertion operation will be done by many participants and the file will keep storing the new feedbacks. To avoid the previous feedbacks are deleted by the new insertion and new storing into the file, append mode will have to be applied.

**2.13. main**

**2.13.1.** First of all, include all the libraries that will be used in the code so that we can make use of the build in functions in the libraries. Besides that, include also all the .h file where we design the functions.



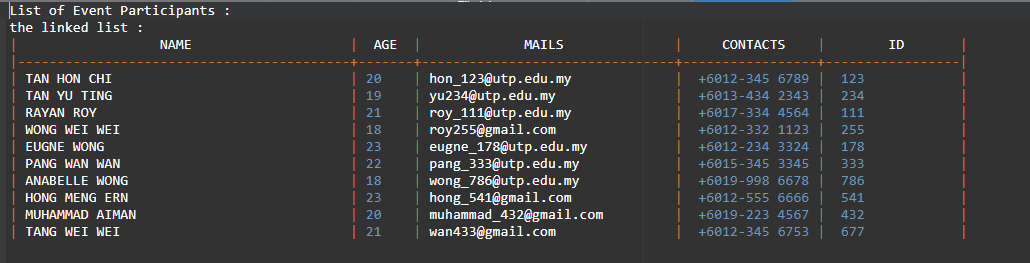
**2.13.2.** Create object of the classes type according to to the .h files in order to call the function that designed in the files. For example, event participants which means that event is the name of class in Event.h where the participants is the object created from the class and will be used to call the functions in Event.h such as participants.lucky\_number();



**2.13.3.** Prompt the user on the option of operations and give the small introduction of each operation/case. Create a switch case that will call the functions according to the user’s input. A while loop is used so that the user can select multiple option and quit by input –1 as option. The keyword “break” is important after each case definition so that the program will only execute the case chosen by user.

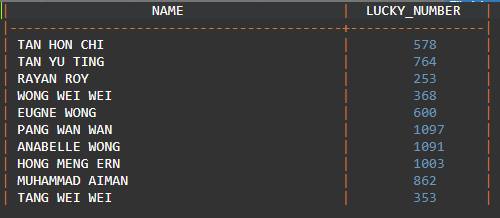
1. **SCREENSHOT OF OUTPUT**

**3.1. Screenshot of Biodata File.txt**



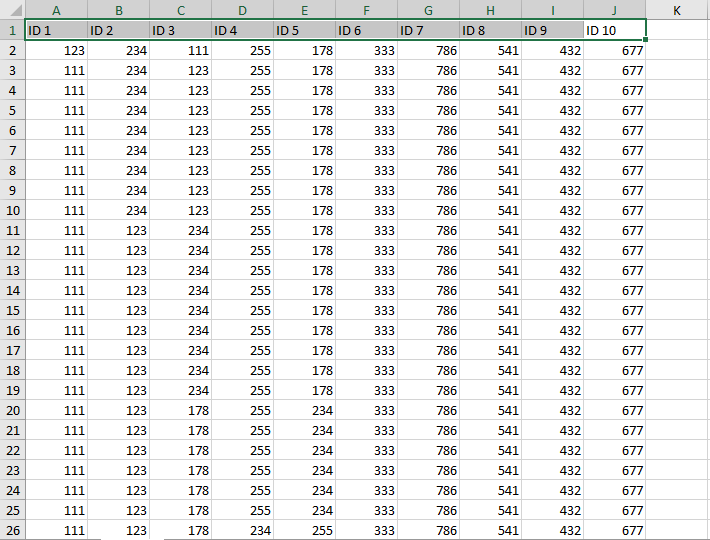
This is the output of Biodata File.txt which has contained the participants’ name, age, mails, contact number and id number. This file is called by savefile\_linkedlist() which is formed in linkedList.h file. Besides, the table of the biodata is fixed by having the functions which are setw() and setfill() that are applied from the iomanip library. Therefore, the length of the inputs will not able to affected the biodata table.

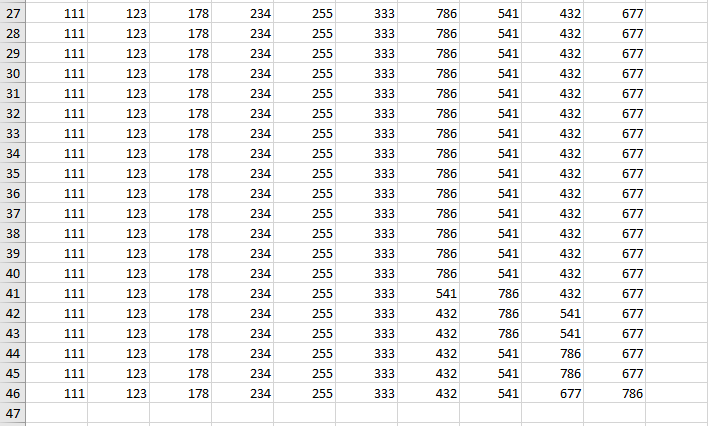
**3.2. Screenshot of Lucky Number File.txt**



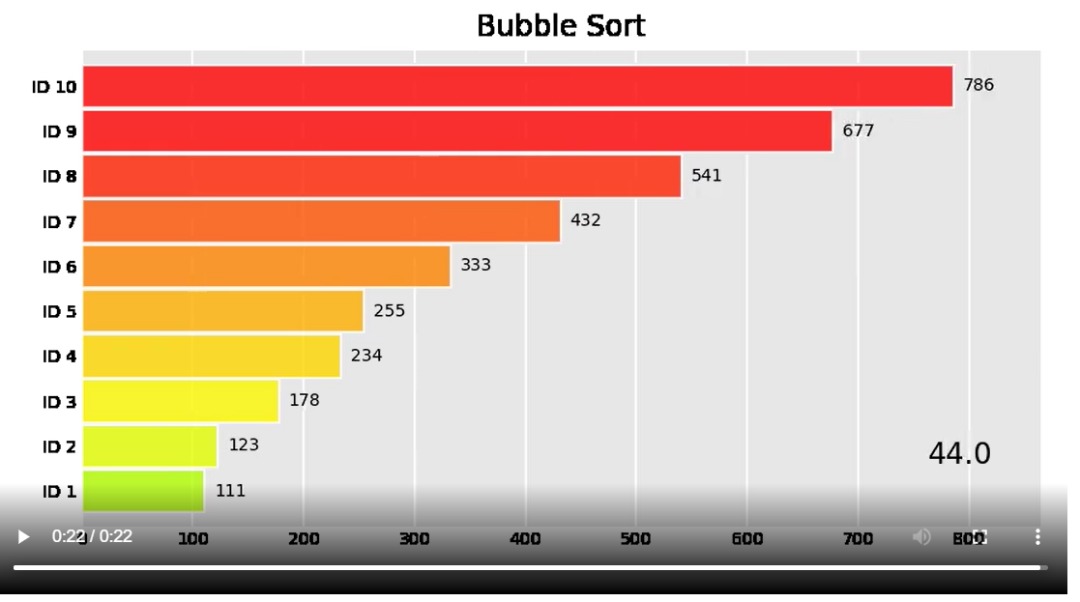
This is the output of Lucky Number File.txt which the lucky numbers are randomly assigned to the participants by using the rand() calculation. This output display is created by lucky\_number() which is done in Event.h file. By having the setw() and setfill() that are used from iomanip library, the length of the inputs will not able to affect the table lucky number and the table is fixed by the declared numbers of width of the setw() and setfill().

**3.3. Screenshot of Bubblesort\_data.txt and Graph Animation of Bubblesort\_data.txt**



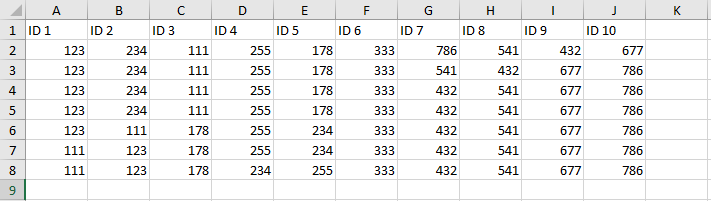


This is the excel data which is named as BubbleSort\_project.

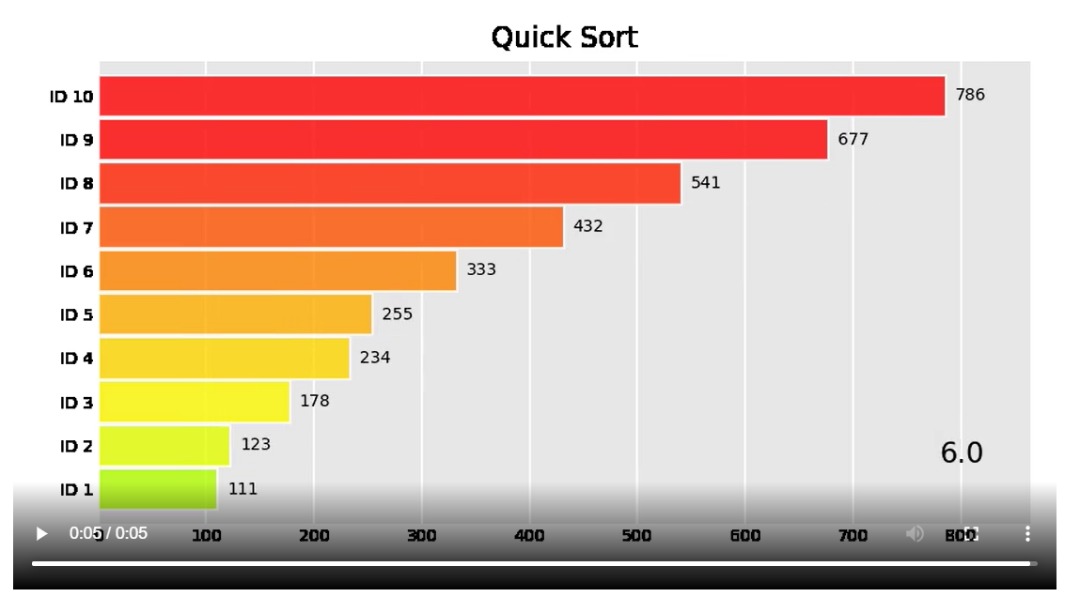


This is the sorting process which has the 10 inputs of id number and it is called by the bubble\_id() in Event.h file. For bubble sorting process, it takes more time to complete the sorting process because the time complexity of O (n^2) is quite big and it is not efficient and suitable to sort for higher data.

**3.4. Screenshot of Quicksort\_data.txt and Graph Animation of Quicksort\_data.txt**

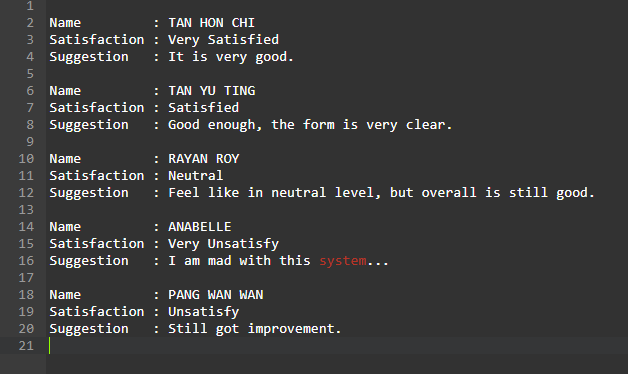


This is the excel data which is named as QuickSort\_project.



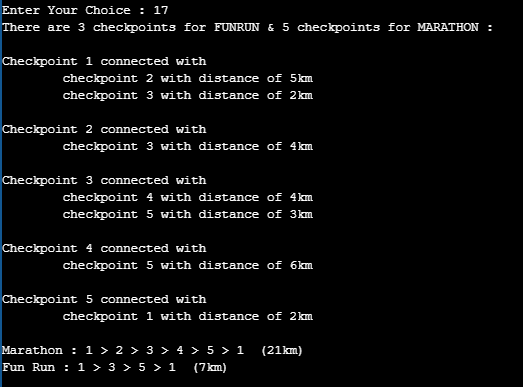
This is the sorting process of 10 id numbers which are done by quicksort algorithm. It has the faster sorting process compare to bubble sorting algorithm because it has the best sorting performance for more items of lists. Besides, the sorting process for quick sort is done in place, there is not requiring additional storage for it.

**3.5. Screenshot of Feedback File.txt**



This is the output of the Feedback File.txt which will store the inputs of name, satisfaction of registration form and suggestions that are done by participants.

**3.6. Screenshot of output Graph.h**



This is the output display which is done by the Graph class in Graph.h. We have 3 checkpoints for FunRun category while 5 checkpoints for Marathon category. Each checkpoint will be the vertices of the graph while the distance between one checkpoint to another checkpoint is the weight of the graph.

1. **TIMELINE WORK**

**10.3.2021**

**7.3.2021**

**6.3.2021**

**5.3.2021**

We have started our first group discussion through Microsoft Teams (MST) and decided the first work distribution of the project.

First work distribution report is done by all the members through our discussion and the complete report file will be submitted to lecturer.

**14.3.2021**

**16.3.2021**

**12.3.2021**

Tan Yu Ting will work and solve for the name spacing problems by adding the getline(cin, name, ‘\n’);

Tan Hon Chi works for solving the phone number spacing problems by changing the int declaration to string declaration and this is able to include the symbols ‘+’ and ‘-’.

At the afternoon time, print() function format have been adjusted and changed by Tan Yu Ting and Tan Hon Chi. Therefore, the output display will be looked better and the biodata table is able to be fixed and not be affected by different length of input.

Rayan Roy has done for the insert functions, print participant function and delete function. However, there are some difficulties on doing the insert name and phone numbers that are with having the spacing and phone numbers.

**17.3.2021**

**18.3.2021**

**16.3.2021**

By having the work cooperation from Rayan Roy and Tan Hon Chi, feedback() function has been completely done.

Tan Yu Ting has created a file to save the biodata (arrays) input by the participants.

Rayan Roy has done for the lucky\_number() function and other members gave their opinions on improving the function.

Tan Hon Chi has also completed the feedback() function at the same day, but just left the file coding part not yet finished.

At the evening time, messages for new work distribution have been decided.

Rayan Roy and Tan Yu Ting will complete the part of frequently\_asked\_questions() function.

Tan Hon Chi will work for feedback() function and organizerInfo() function.

In the same day, organizerInfo() and Frequently\_asked\_question() functions are completed and improved by all of us.

Besides, we have our second group discussion which is for deciding the event title and this discussion is done through Whatsapp group chat.

**19.3.2021**

**24.3.2021**

**22.3.2021**

We have the third discussion for next functions that will need to be completed into this project.

Work Distribution :

Tan Hon Chi will work for the uppercase function which is used to change the lowercase letters to uppercase letters, add Rules and Regulation information into Frequently\_asked\_question() and insert time taken function.

Rayan Roy will have to complete lucky\_number() function convert the linked list into array and implement it in Event.h. New functions convertArr() and findlength() are designed to get the names of participants from linked list to array. Functions getName and getSize also created.

Tan Yu Ting will have to complete bubblesort() function that is done by using linked list. The list of participants will be sorted according their id number. The bubblesort() is applied to linked list.

Linked list coding has been done by Rayan Roy and the proceed work has been continued by Tan Yu Ting.

Rules and Regulations has been added into Frequently\_asked\_question() function and done by Tan Hon Chi.

Then, we complete these works through MST meeting whenever we are free. We convert to save the biodata of participants from using array to linked list. Then, we edit the available function designed before this to be applicable for linked list. For the insert participant function, the pointers will be representing participants where the data is saved in the attributes. Create a new function named find() to find the position of participants by using the id input. Edit the function remove\_position() to be applicable for linked list where it will remove the participant according to the position input. Create a new remove\_id() function to remove the participant according to the id input. Edit the two functions named edit\_int and edit\_string to allow the user edit their biodata that saved in the attributes of the pointer in the linked list.

**25.3.2021**

**27.3.2021**

**26.3.2021**

All the members have involved in doing the work of visual illustration which is the python animation.

Working on this, research work will be the first work process which is to get more information and knowledge on python animation. We will share the information to each other and learn the working on python animation from each other.

Next, coding in python graph animation will be continued by all of us.

File for biodata is done together by all of us through the meeting discussion. Besides, we have also involved in problems research so that more details and more ideas will be able to get.

Tan Yu Ting – Sorting for time inserted using array.

Tan Hon Chi and Rayan Roy – Complete the works for insert time taken function and uppercase method.

**30.3.2021**

**28.3.2021**

**29.3.2021**

We have our fourth discussion which is done through Microsoft Teams (MST) platform to discuss on the functionality that required graph processsing and finally we have decided to design a graph that displayed the marathon route. Then, Graph.h is created and the graph is designed by using addEdge function.

Next, final report which contains the information and discussion of the project coding has been done by all of the members. We have discussion which is for discussing the report format so that we have an organized report.

We have our last meeting through Microsoft Teams (MST) which is for recording the group presentation of our project.

Next, after completing the presentation recording, Rayan Roy will start the editing work so that the video will look better and organised.

Tan Hon Chi – Input 10 participants' biodata and save the important data required for animation into file for example the Bubbesort\_data.txt and Quicksort\_data.txt

Tan Yu Ting – Save the data obtained from the two file and convert into Excel file. Then, upload the two Excel data file into Jupyter notebook and design the animation code which is the bar chart race to display the animation to sort the list of number which is the participants' ID.

Rayan Roy – Design the BST.h class that make use of the knowledge of binary search tree where it will input the ID data into binary tree and the user can search for the existence of a participant's ID through the binary search tree.

**31.3.2021**

Submission Date.