SCHOOL OF COMPUTING

FACULTY OF ENGINEERING

SECJ2013-06 STRUKTUR DATA DAN ALGORITMA (DATA STRUTURE AND ALGORITHM)

MINI PROJECT:

HOSTEL MANAGEMENT SYSTEM

Group Name: Group 8

Members:

Name	Matric No
Aum Jeevan A/L Aum Nirangkar	A20EC0017
Brendan Dylan Gampa anak Joseph Dusit@Dusit	A20EC0021
Hafiy Hakimi Bin Saiful Redzuan	A20EC0040

Lecturer's Name: Ts Dr Johanna binti Ahmad

Submission Date: 27th JANUARY 2022

Video presentation link: https://youtu.be/pk28oloBbe0

1. INTRODUCTION

1.1 Synopsis Project

In mini project, our group has been assigned to create a Hostel Management System. The designed system is made specially for student book their rooms in college and admin to view the student details of the room booking.

The system has implemented data structure concepts such as linked list, quick sort, stack, searching and queue. Each concept is applied to perform different functions on the system. There are a total of 10 classes that we have implemented such as login, college registration, check in, payment and view student details. Before the students can select their desired room number, they need to select which college they want to enroll in as well as including their personal information. Room number is based on the format of wing building, floor and then the number. Staff is allowed to sort, delete, and view the data of student details. As for students, they can register their self-details, room booking, and retrieve receipt to pay the rents. The stack implementation is based on managing the room number where the staff can delete the past check in and check out dates of the room details. The queue meanwhile is to view and delete past student's personal details based on the FIFO (First In First Out) concept.

1.2 Objectives of the Project

- Provide centralized hostel management system.
- Provide staff with an easier view of booking details done by students.
- Prevent data loss by storing students' booking details in the system.
- Allow an efficient engagement from the students to navigate through the system while input necessary data for hostel booking.
- Enable staff to manage the check in and check out dates of the room booking.
- Provide staff with an easier algorithm of view and delete student personal details.

2. SYSTEM ANALYSIS AND DESIGN (USE CASE, FLOWCHART AND CLASS DIAGRAM)

2.1 System Requirements [CLASS DIAGRAM]

The system target groups are students and staff (admin).

Target Group	Tasks
Students	Students are in charge to fill in personal details, choose desired college, select room number as well as check in and check out dates. Students will be given total amount to be paid for room renting and an automated receipt that display their details. Students are also eligible to view the details inputted earlier by key-in their matric number.
Staff	Staff is responsible to view the whole student's details and perform sorting on student's matric number. Staff can also view the details of a single students by inputting student's matric number. The staff has the rights to view and delete the past room details which includes the student's personal details.

Table 1: Target groups with respective tasks

The system implements 10 main classes with respective purposes based on the class diagram in Figure 1.

Class	**Task**
Login()	Students and staff login into the system by entering their username and password.
RegistrationCollege()	Students fill in personal details and select desired college.
CollegeRoom()	College room list displayed, and students choose room number.
CheckInOut()	Students set check in and check out dates which includes days interval in between.
Payment()	System set total payment based on the days stayed.
Receipt()	Display automated receipt to students.
StaffView()	Staff view all student list at output data file and sort matric number.
StaffSearch()	Students and staff search a student's details via matric number.
RoomStack()	Staff view and delete past room details.
MatricQueue()	Staff view and delete student's personal details.

Table 2: Classes implemented in the system with respective tasks

Class Diagram

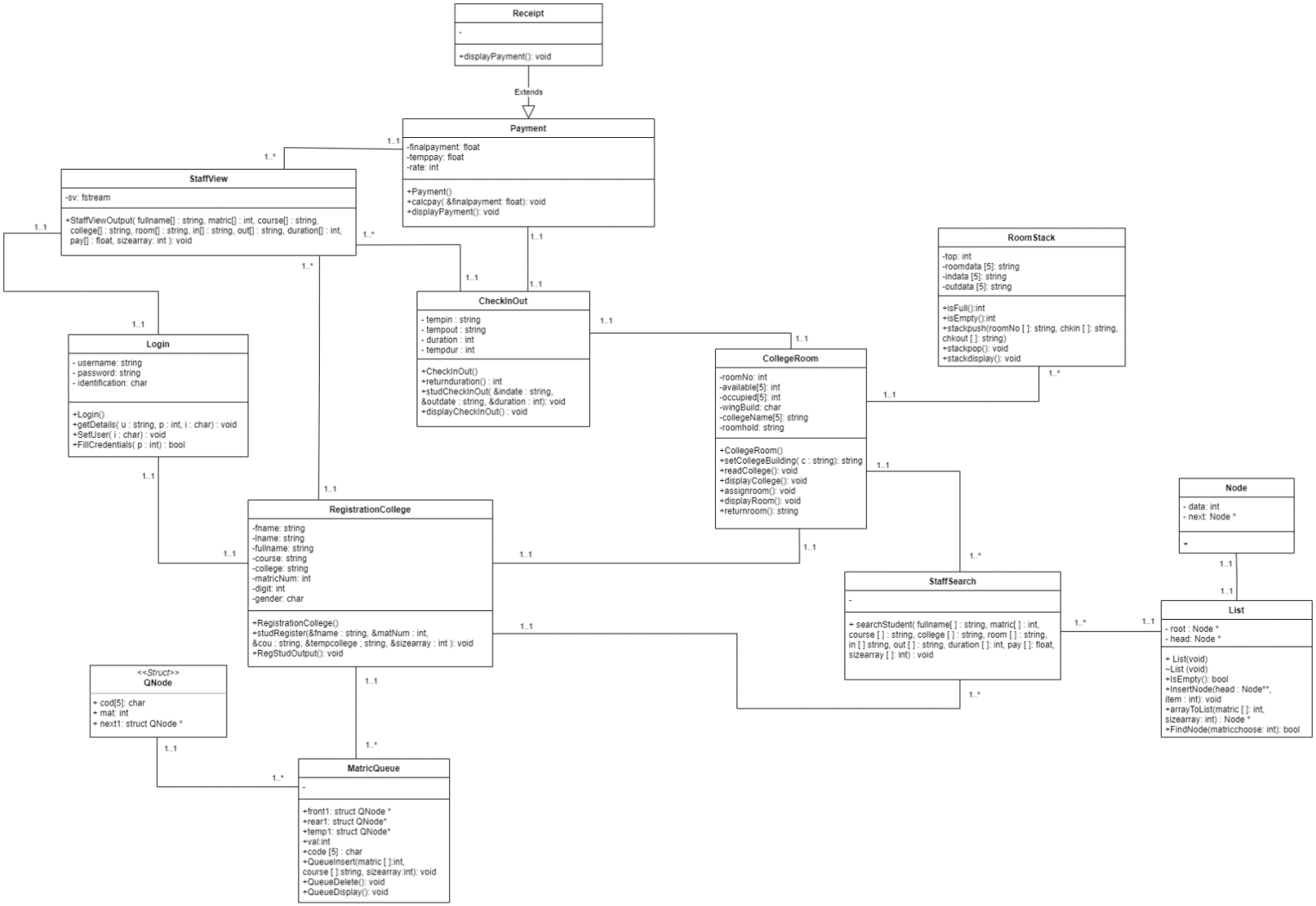


Figure 1: Class diagram of Hostel Management System

2.2 System Design [FLOWCHART]

Flowchart 1: Login (Filled by students)

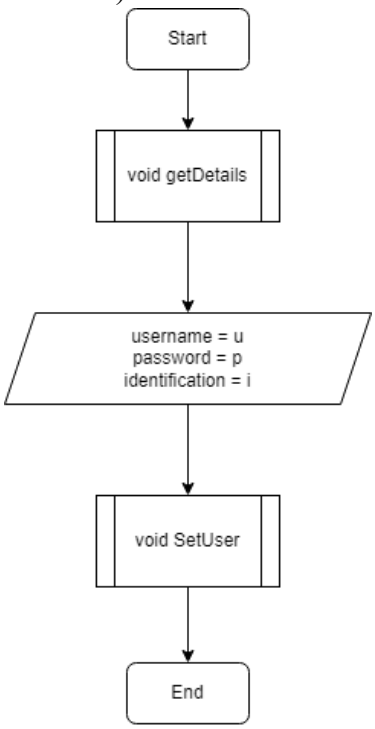


Figure 2: Flowchart for Login

Above flowchart is for students to log into the system. It will prompt the students to fill in their username, password and identification through void getDetails() function. Next, the system will proceed with void SetUser() function according to their identification.

By: Hafiy Hakimi bin Saiful Redzuan

Flowchart 2: Student registration (Filled by students)

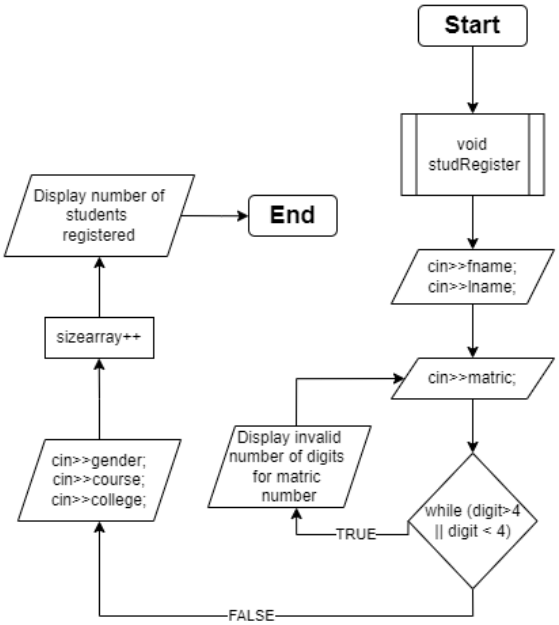


Figure 2: Student registration flowchart

The data structure applied for this flowchart is array only. The class RegistratiCollege functions to get input from students about their name, matric number, gender, course, and college desired. By having each item are inputted correctly, the array size will be incremented by and notify the number of students registered. Validation process occur on the matric number where the system validates the number of digits for matric number.

By: Brendan Dylan Gampa anak Joseph Dusit@Dusit

Flowchart 3: College room registration (Filled by students)

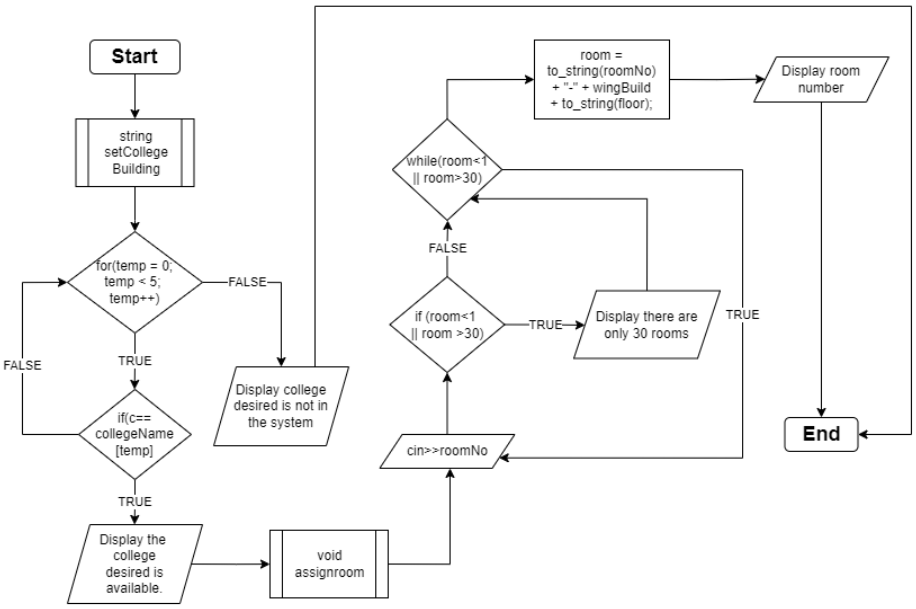


Figure 3: Flowchart for college room registration

This flowchart only applies array for the data structure. This is because this flowchart requires to determine the room number of students based on their desired college. If their desire college name is available in function string setCollegeBuilding(), it will prompt user to input their desired room number in function void assignroom(). Since there are only 30 rooms available for each college, room number can only be included within that range. If user input other integer beyond 30, system will prompt error message and user needs to key-in the room number again. There are no limits for the validation. Once room is selected, system will display the room number based on the floor and wing building where the room is located.

By: Brendan Dylan Gampa anak Joseph Dusit@Dusit.

Flowchart 4: Check-in and check-out (Filled by students)

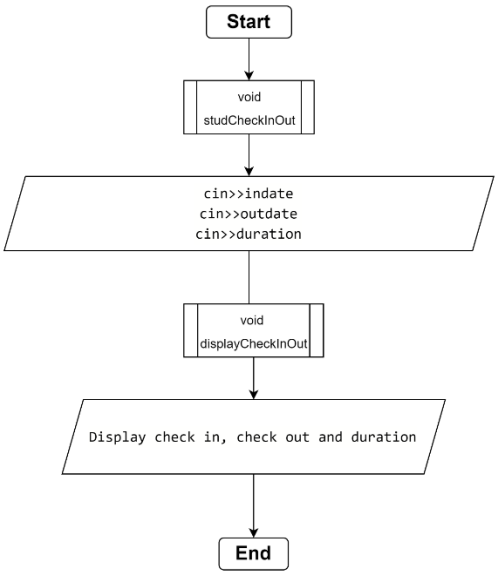


Figure 4: Flowchart for check-in and check-out

The flowchart above aims to provide the check in, check out and duration details for the student. It starts off by prompt user to input the check in date, check out data and duration in the void studCheckInOut(). Afterwards, it will go to void displayCheckInOut() where it will display the check in date, check out date and duration of stay in a well-organized way.

By: Aum Jeevan A/L Aum Nirangkar

Flowchart 5: Payment (Filled by students)

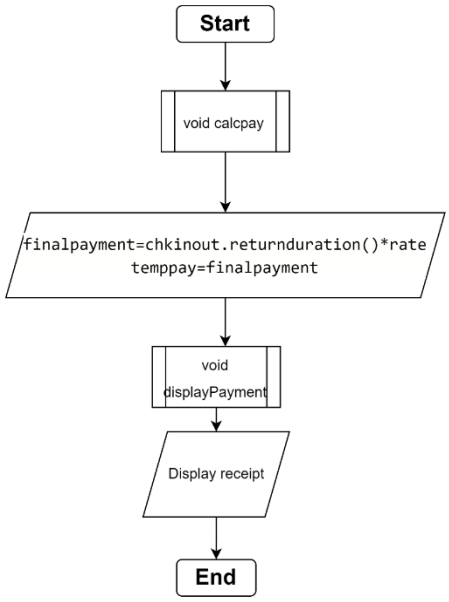


Figure 5: Flowchart for payment

The flowchart above aims to calculate the payment needed to be done by the student and display the receipt. It starts off in void calcpay() where it will calculate the payment needed by using the formula finalpayment=chkinout.returnduration()*rate. Afterwards, in void displayPayment(), it will display the receipt to the user. The receipt states the rate per day as well as the final amount needed to be paid.

By: Aum Jeevan A/L Aum Nirangkar

Flowchart 6: Receipt (Filled by students)

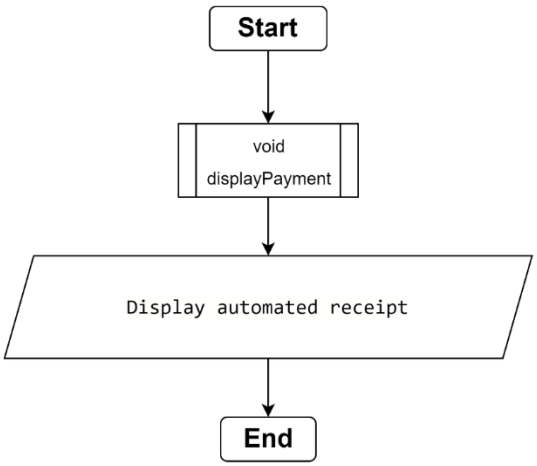


Figure 6: Flowchart for receipt

This flowchart aims to display the automated receipt. In void displayPayment(), it will display the information such as registered college, college room, check in and check out details, payment details, and so on. This helps the user to see all the output from other processes in one receipt. Thus, helping them reassure no errors taking place.

By: Aum Jeevan A/L Aum Nirangkar

Flowchart 7: Staff View (Accessed by staff)

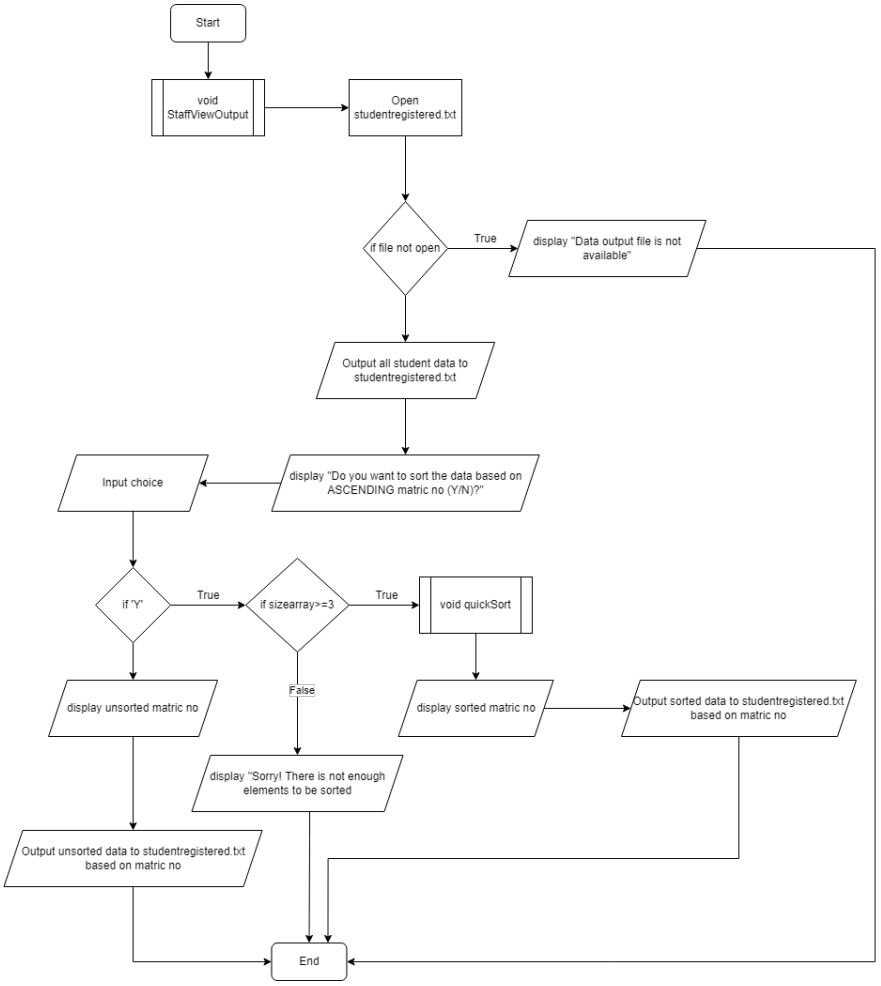


Figure 7: Flowchart for Staff View

The flowchart above shows the display function for staff. It will start with void StaffViewOutput() function that will prompt the system to open studentregistered.txt file. Next it will go through a condition where if the file didn't managed to be opened, the system will display "Data output file is not available" while if the file is opened, it will proceed to output all data into the file and display "Do you want to sort the data based on ASCENDING matric no (Y/N)?" which prompt the staff to give an input. If the staff wants the data to be in ascending order and if the data is more than 3, it will run void quickSort(), display sorted matric no and output the data into student file. Otherwise the unsorted data will be displayed and kept in the student file.

By: Hafiy Hakimi bin Saiful Redzuan

Flowchart 8: Search specific student details based on matric number. (Accessed by Staff)

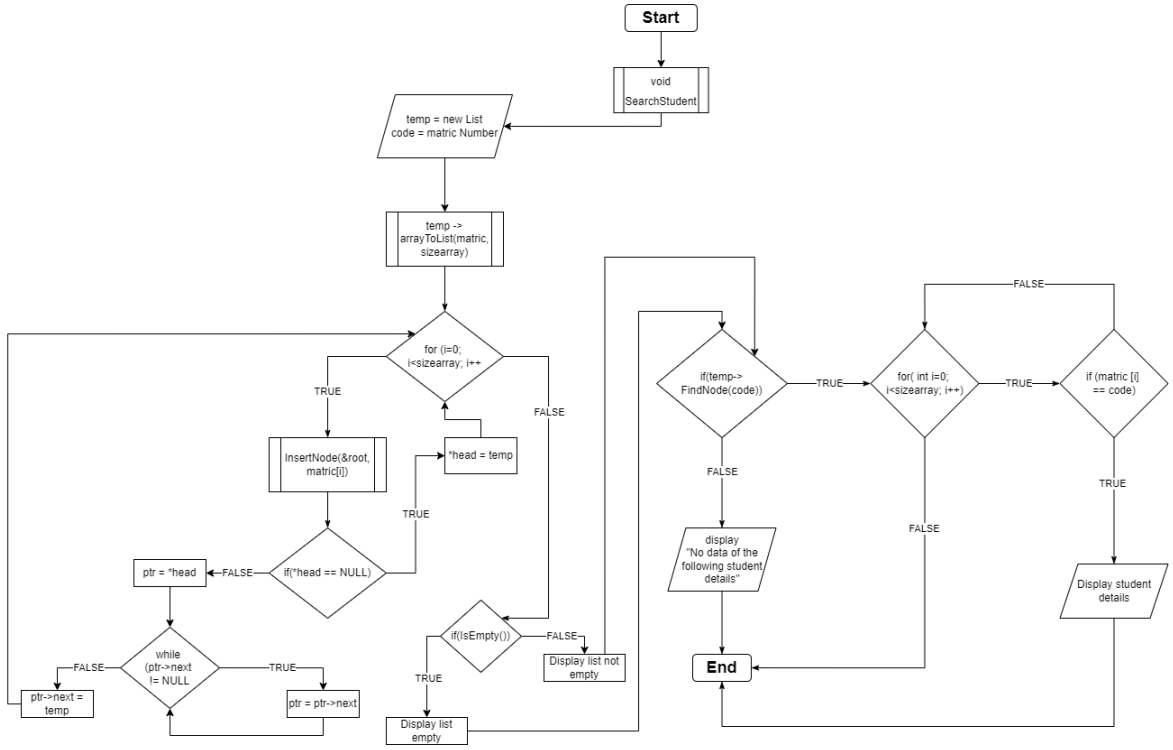


Figure 9: Flowchart for staff search

The data structure applied for this flowchart is **singly linked list**. The arguments of students details are passed from main() to void SearchStudent in class StaffSearch. This class functions to search a single specific student's details by matric number. Elements in the array of matric number are sent to the calling function arrayToList() which passed each element of matric number into the nodes. Every time the elements are being inputted into the node, the pointer ***ptr*** points to ***next*** pointer to input the next element. Once node insertion is completed, the system goes through IsEmpty() function which displays that the linked list is not empty if there are nodes in the linked list. Function FindNode() is evaluated by comparing the desired matric number with the matric number in the linked list. If the result is true, then the program will display the student details based on the student's matric number. Else, an error message will be prompted.

By: Brendan Dylan Gampa anak Joseph Dusit@Dusit

Flowchart 9: View and delete room details with check in and check out data. (Accessed by Staff)

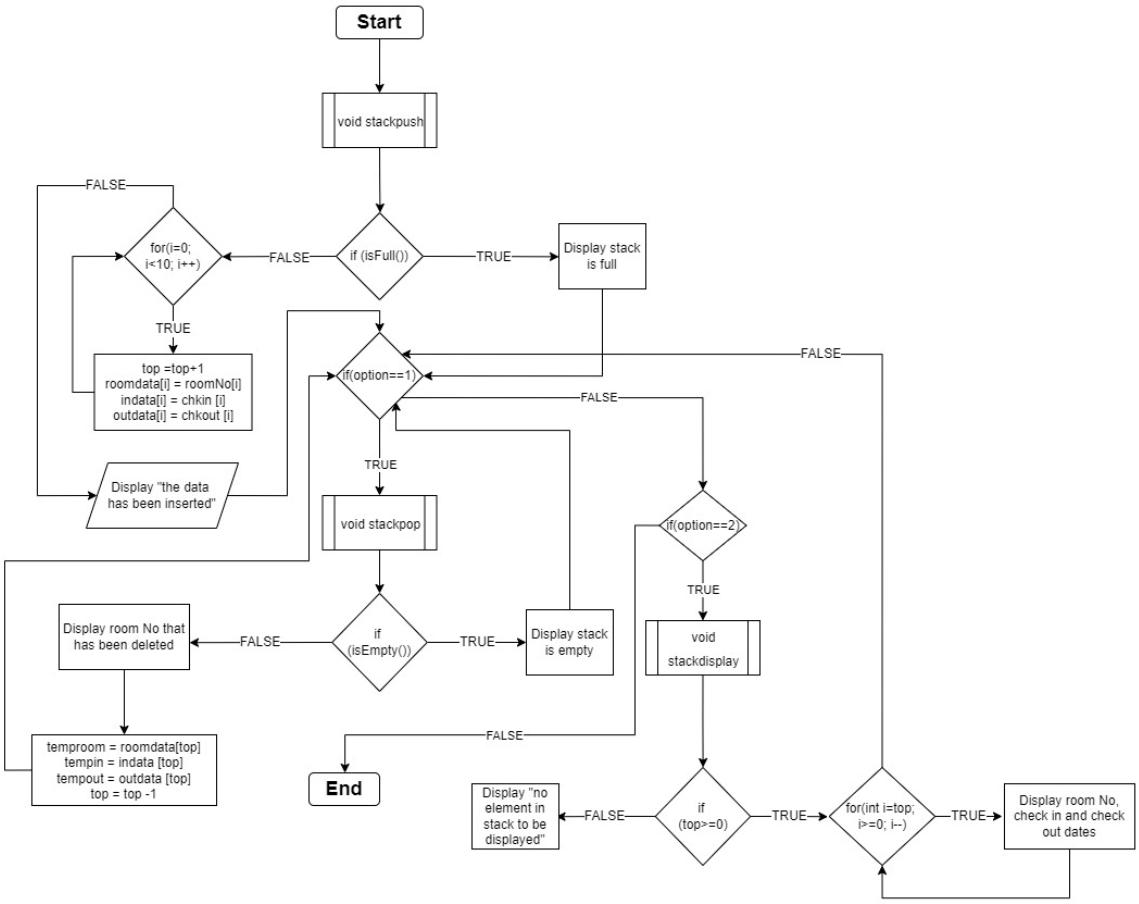


Figure 10: Flowchart for manage room details

The data structure applied for this flowchart is **stack**. The array arguments of room number, check in and check out dates are passed from the main() into the void stackpush() in class RoomStack. This class functions to view the room details with its check in and check out dates. By implementing LIFO (Last In, First Out), the last elements will be at top. If the condition isFull() is false, then data from the array will be inputted as top is incremented by 1. Option 1 means to delete the data at the top through function stackpop(). If staff selects this option, program will delete the elements at top, which is the last element that has been inputted. At the same time, the value top will be decremented by 1. Else if one selects option 2, program will display the remaining data after the deletion and sorted based on LIFO concept through function void stackdisplay(). The program exits and return to the main menu if options selected are not equal to 1 or 2.

By: Brendan Dylan Gampa anak Joseph Dusit@Dusit

Flowchart 10: Insert matric number and course code. (Accessed by Staff)

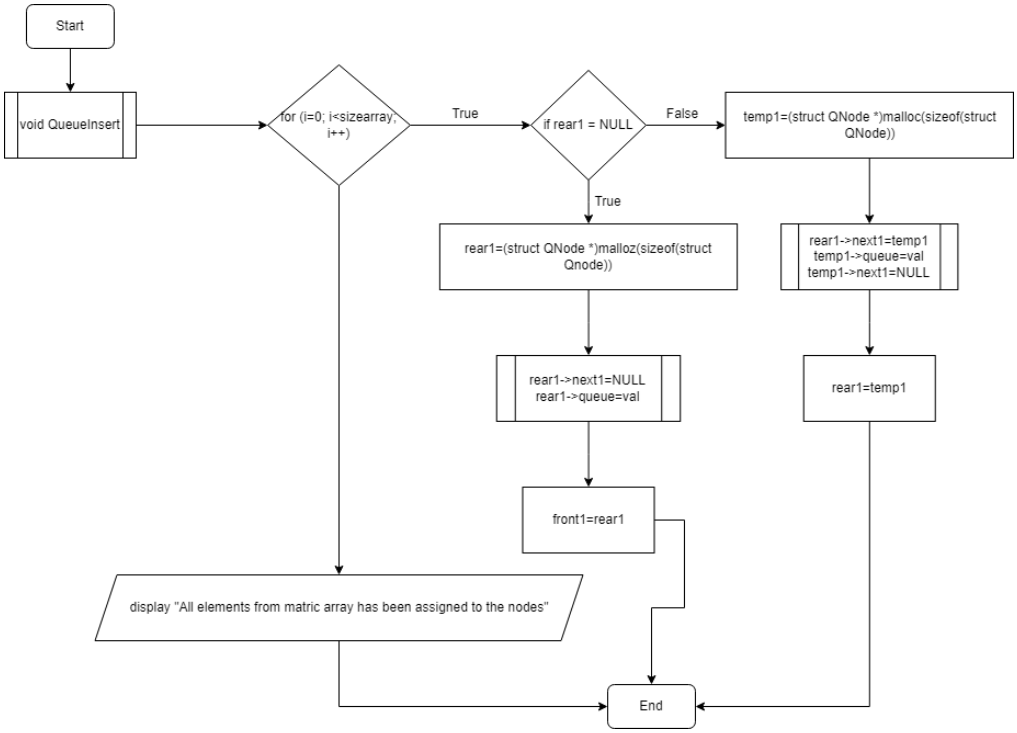


Figure 11: Flowchart for Insert Matric Number and Course Code

The above flowchart shows the flow for insert matric number and course code that can be accessed by staff members. The data structure for this function is queue. It begins with void QueueInsert() function and through a for loop to assign nodes to the matric numbers. While the for loop is still true, it will check if rear1=NULL, if it is true then the matric number will be assigned to rear1 and if it is false, it will be assigned to temp1. However if the for loop is false, it will display “All elements from matric array has been assigned to the nodes”.

By: Hafiy Hakimi bin Saiful Redzuan

Flowchart 11: Delete and display matric number and course code. (Accessed by Staff)

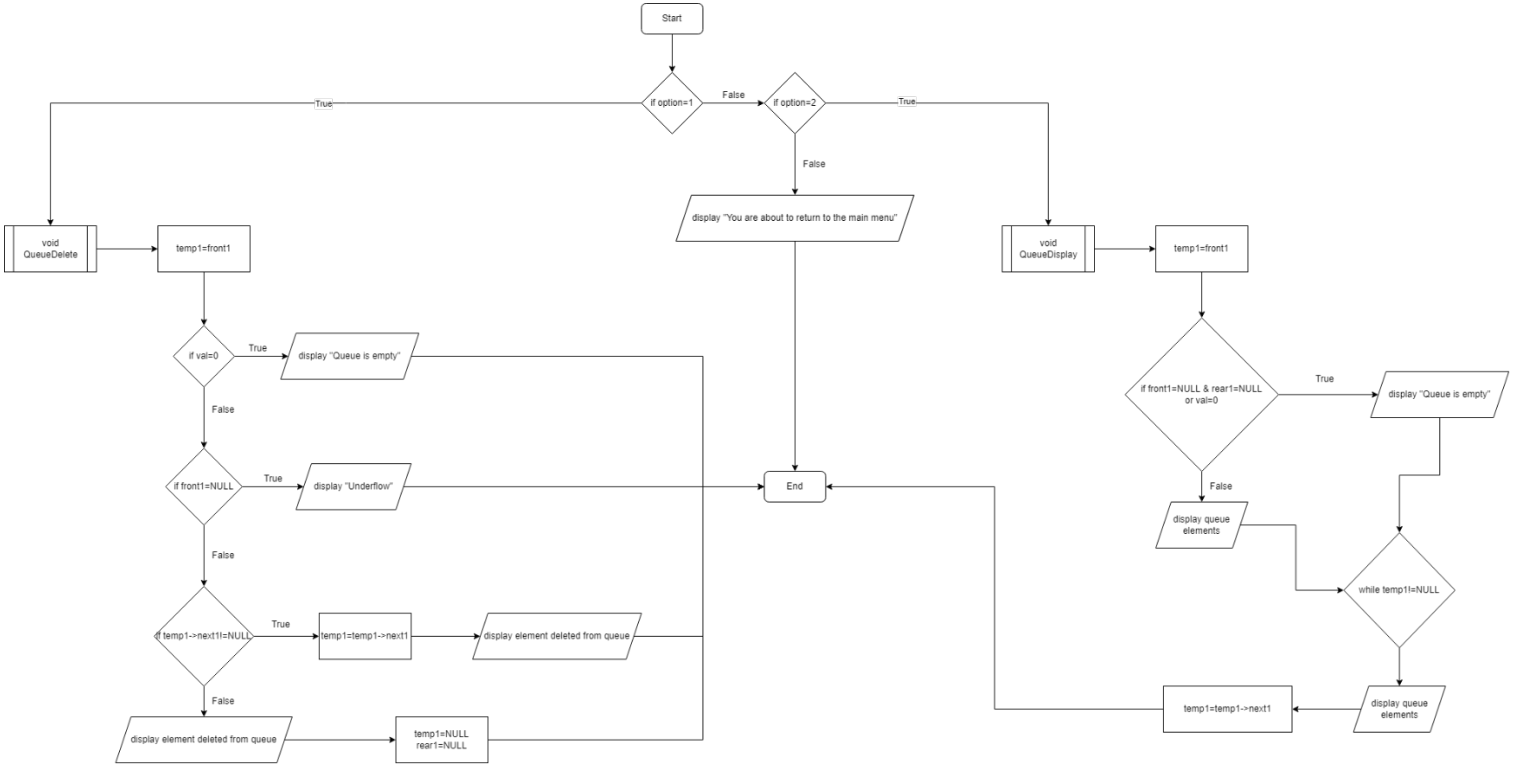


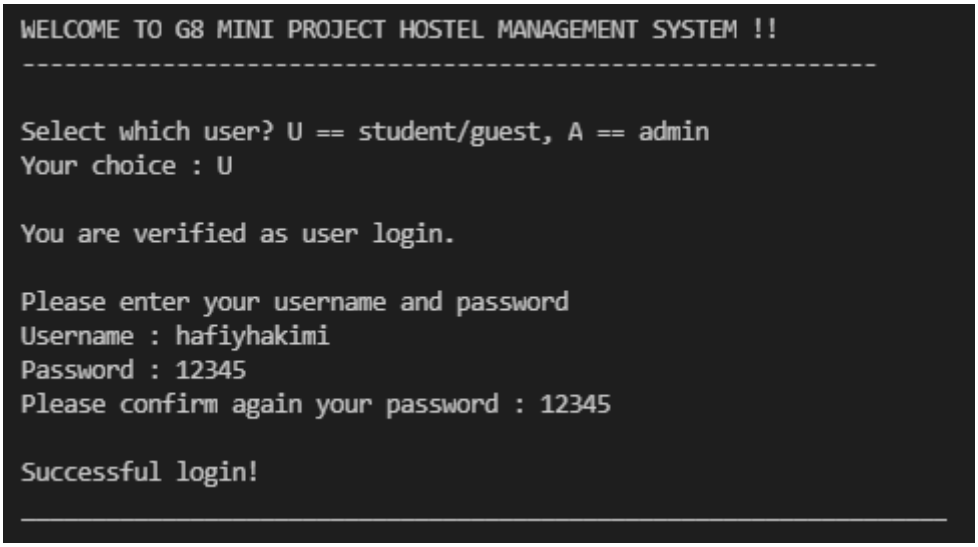
Figure12: Flowchart for Delete and Display Matric Number and Course Code

Above flowchart is based on queue data structure. Functions QueueDelete and QueueDisplay is designed to remove and display matric number and course code from the queue. It uses the same concept of dynamic memory allocation for queue delete and for display only designed to display the data.

By: Hafiy Hakimi bin Saiful Redzuan

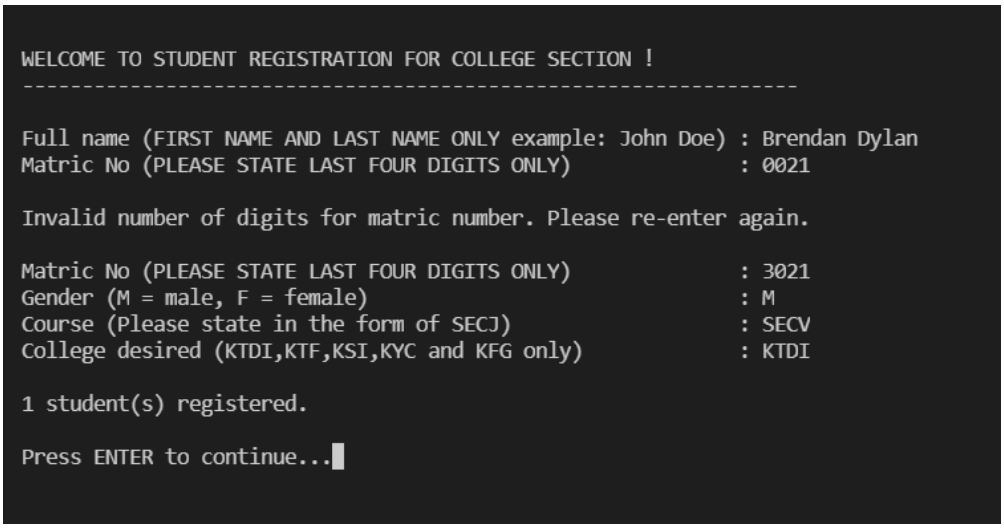
3. SYSTEM PROTOTYPE

LOGIN



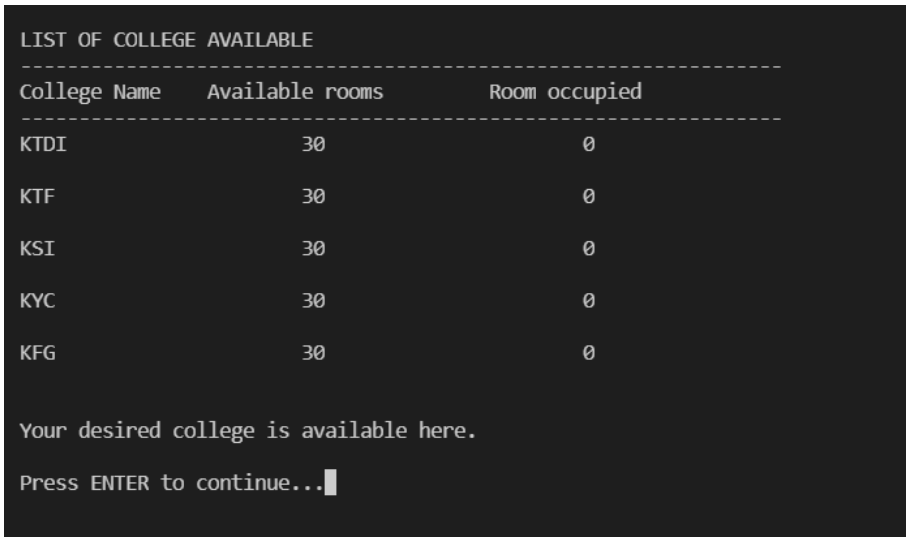
This screen shows the login function for the system. At first the system will ask the user to choose whether they are students or admins to give access. Once they are verified, they are prompted to fill in their username, password and they are asked to reconfirm their inserted password. Once the process finished, the system will display successful login and will display the next interface.

STUDENT REGISTRATION

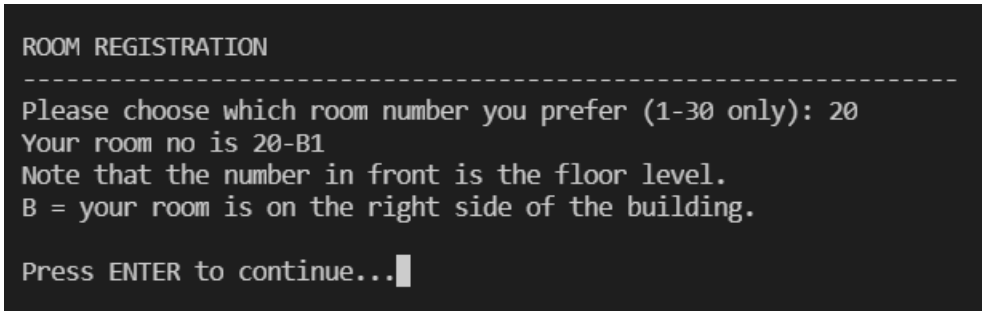


The screen shows the student registration for students to fill in the details required. Details such as name, matric number, gender, course, and college desired. We have also included validation for each item that user need to input. For instance, when user enter the matric number as 0021, the system takes the value as 2 digits only and notify user that one need to re-enter again. There are limits of validating the items as long as user input what the system wants. Once the details have been filled in, system will notify how many students have been registered.

COLLEGE ROOM

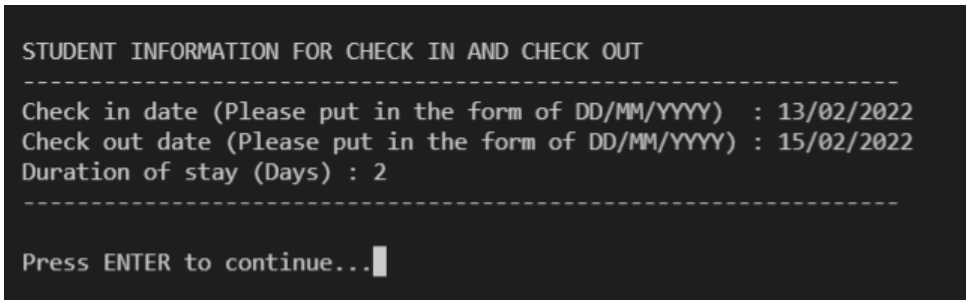


The screen shows the list of college name together with the number of rooms available and occupied. The interface will only prompt this as to notify user how many rooms are available at their desired college.



Once the room college list has been displayed, the system then required user to input desired room number. Once selected, system will determine which floor that one's room would be and also the wing building on each floor.

CHECK IN AND CHECK OUT



Upon coming to check in and check out, the screen will display a list of input statements. The user will start off by inputting his check in date followed by check out date and ending with the duration of his stay. Here, we can see that the user will check in on the 13/02/2022, check out on 15/02/2022, and the duration of the stay is 2 days

PAYMENT AND RECEIPT

```
AUTOMATED RECEIPT
-----------------------------------------------------------------
Full name   : John Wick
Matric No.  : 3455
Gender      : M
Course      : SECV
Room No     : 17-B1
-----------------------------------------------------------------
Check in date       : 13/02/2022
Check out date      : 15/02/2022
Duration of stay    : 2 days.

Price rate per day : RM 2
Please pay your final amount RM 4.00 at your block office
-----------------------------------------------------------------
THANK YOU !! ENJOY YOUR STAY <3

Do you want to relogin? (Y = yes / N = no)
Your choice :
```

After completing student registration, choosing his desired room as well as check in and check out, the user will arrive at the payment and receipt screen. Here, an automated receipt is displayed to show all the details of the user. Besides that, it also shows the price rate per day and the final amount needed to be paid at a specific location. In this case, we can see that the price rater per day is RM 2 and the amount needed to be paid is RM 4.00 at the block office.

VIEW STUDENT DETAILS AND SORT MATRIC NUMBER

```
STUDENT'S REGISTERED

No.  Full name        Matric   Course  College Registered  Room No  Check In Date  Check Out Date  Duration  Payment required (RM)
1    John Wick        1200     SECV    KSI                 17-B1    13/02/2022     15/02/2022      2         4.00
2    Lisa Surihani    2011     SECJ    KTF                 19-B1    7/02/2022      10/02/2022      3         6.00
3    Haziq Aqil       3221     SECJ    KTDI                9-B0     8/02/2022      11/03/2022      31        62.00
4    Rohit Jarjit     3455     SECF    KYC                 11-A1    9/02/2022      13/02/2022      4         8.00
5    Siew Yuen        7899     SECD    KTF                 16-B1    13/01/2022     20/01/2022      7         14.00


-----------------------------------------------------------------
Sorted matric no : 1200 2011 3221 3455 7899
-----------------------------------------------------------------

PROBLEMS 2  OUTPUT  TERMINAL  DEBUG CONSOLE
Do you want to sort the data based on ASCENDING matric no(Y/N)?
Your choice : Y

Do you want to relogin? (Y = yes / N = no)
Your choice :
```

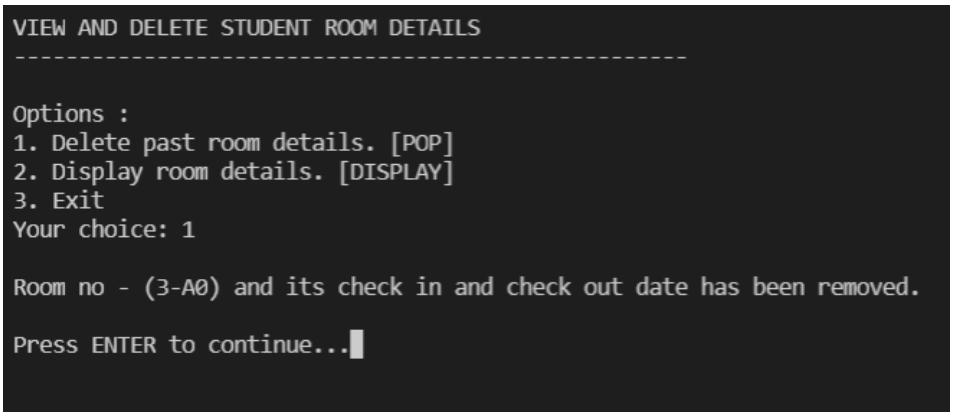
This interface shows all the registered student details based on the registration made by the students themselves. This interface is only accessible by admins or staff members. The details are also sorted in ascending order of their matric numbers as the staff or admin were prompted to choose whether they want it to be sorted or not. After all the processes have finished, the system will prompt the admin whether they want to relogin or not.

STAFF SEARCH

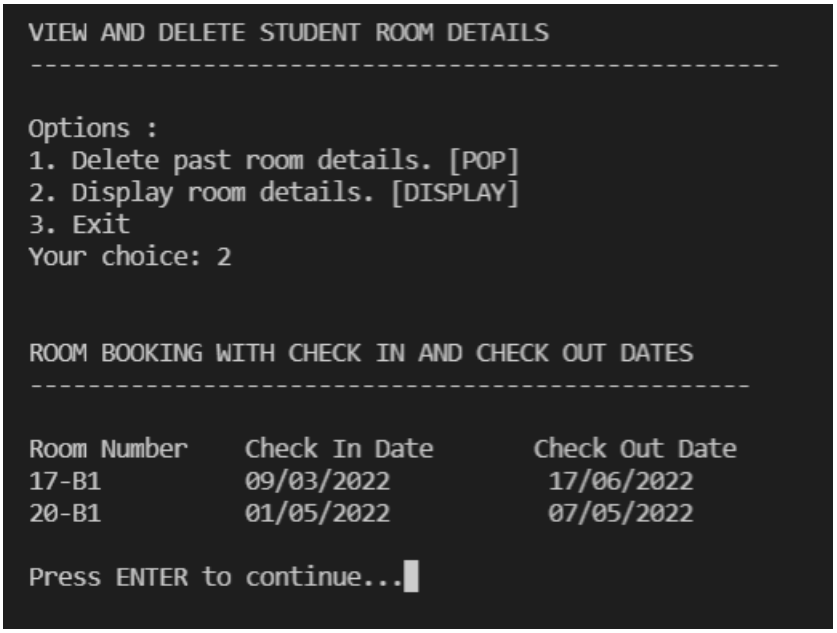
```
Select the following options
1. Student Registration (STUDENT ONLY)
2. College Room Status [Includes room booking and payment] (STUDENT ONLY)
3. View Student's booked room. (STAFF ONLY)
4. Search Student's Details (BOTH CAN ACCESS THIS SECTION)
5. Manage college room. (STAFF ONLY)
6. View and delete student personal details. (STAFF ONLY)
-----------------------------------------------------------------
Your choice : 4

The list is not empty!

Please enter the Student Matric Number: 3021

Matric Number search is available !!

The following student's matric number is available !!
Full Name :  Brendan Dylan
Matric No : 3021
Course    : SECV
College   : KTDI
Room      : 20-B1
Check In  : 01/05/2022
Check Out : 07/05/2022
Duration  : 6 days
Payment   : RM 12.00
```

The screen shows on searching a specific student's details by key in the matric number of the desired. Note that this feature is accessible to both target groups. This is to allow students to double check if their details are correctly stored in the system. As for staff, they can also navigate through the system to find the specific student's details. Once the desired details have been displayed, system will prompt user to the main menu to perform another searching or select other options.

MANAGE ROOM DETAILS

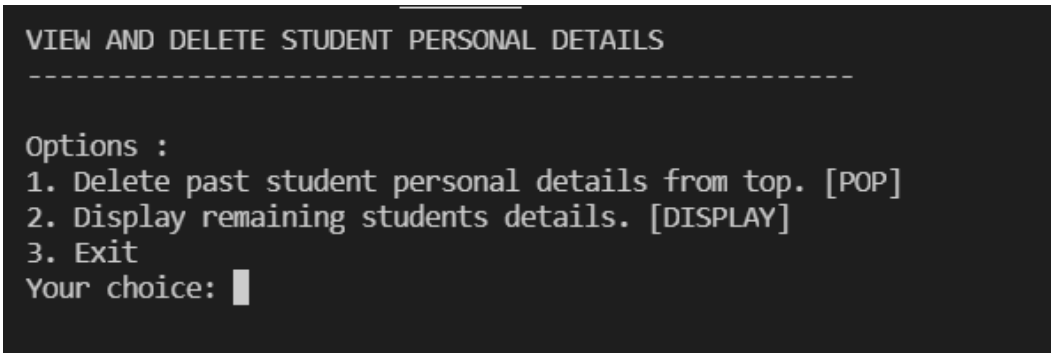


The screen portrays a mini menu to view and delete the student room details. This particular section is created in case the students have past booking of the rooms; the system would preferably delete it to free some memory. When staff selects option 1, the room details at top will be deleted and removed from the system. Staff can select option 1 multiple times until the system prompts a message that the stack is empty.

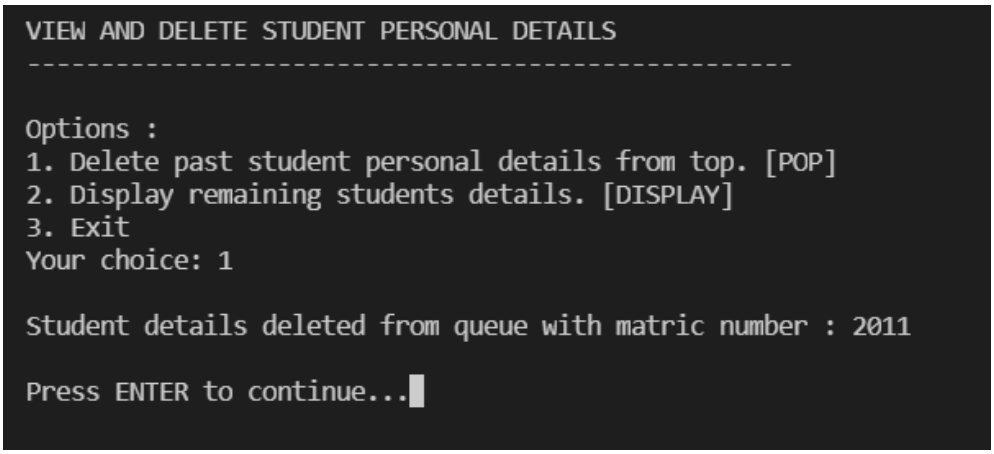


The second screen on the mini menu would be to display the remaining room details in the system if there is any deletion of room details conducted before. As in the first screen related to the deletion process, it is noted that room details for room 3-A0 have been removed. This means it will not be displayed when staff wants to see the remaining room booked in the system. The display function is important to alert the staff if there are any past check out room dates that are yet to be removed.

MANAGE STUDENT PERSONAL DETAILS



The screen above shows the view and delete student personal details under matricqueue function. The system will first insert all necessary data from the other data structures to be inserted into the queue. Once the insert data has finished, the system will ask the admin whether they want to delete past student personal detail or to display all personal detail of each students available and ask the admin to enter their input.



The screen above shows the view and delete student personal details under matricqueue function. This is the system after an admin choose to delete a student details from the top of the queue. The system will display the matric number of the student that is going to be deleted and ask the admin to press enter to continue. Once the admin click enter, the student details will be deleted.

4. DEVELOPMENT ACTIVITIES

Meeting Date	Members Participate in the meeting	Activity	Task for each member	Task Achieved (Yes/No)
18/01/2022	Jeevan Hafiy Brendan	Discuss Improvisation from Assignment 2	Contribute ideas into the discussion.	Yes
20/01/2022	Jeevan Hafiy Brendan	Divide tasks on functions for each member	Volunteer to be responsible for functions of his choice.	Yes
20/01/2022	Jeevan Hafiy Brendan	Write and improvise the coding from Assignment 2 based on the first meeting	Contribute ideas to fix errors in coding. Give ideas on where to implement the data structure concepts.	Yes
21/01/2022	Jeevan Hafiy Brendan	Divide tasks for report writing	Volunteer to be responsible for parts of report of his choice.	Yes
25/01/2022	Jeevan Hafiy Brendan	Finalize coding and report. Assign video parts for group presentation.	Each member has respective parts and concepts to be mentioned in the video. Jeevan compiled all the video parts.	Yes

5. APPENDIX

.cpp coding file

```
//MINI PROJECT HOSTEL MANAGEMENT SYSTEM GROUP 8 SECTION 06 DATA STRUCTURE AND ALGORITHM
//1. AUM JEEVAN A/L AUM NIRANGKAR A20EC0017
//2. BRENDAN DYLAN GAMPA ANAK JOSEPH DUSIT@DUSIT A20EC0021
//3. HAFIY HAKIMI BIN SAIFUL REDZUAN A20EC0040

//LECTURER: TS. Dr. Johanna binti Ahmad

#include <iostream>
#include <string>
#include <stdio.h>
#include <ctype.h>
#include <fstream>
#include <cstdlib>
#include <iomanip>
#include <unistd.h>
#define collegelist 5
using namespace std;

//global object declaration
fstream collroom;

//function to do the partition for quick sort
int partition(int mNo[], int first, int last){
    int pivot, temp;
    int loop, cutPoint, bottom, top;

    pivot=mNo[first];
    bottom=first;
    top=last;
    loop=1;

    while(loop){
        while(mNo[top]>pivot){
            top--;
        }

        while(mNo[bottom]<pivot){
            bottom++;
        }

        if(bottom<top){
            temp=mNo[bottom];
            mNo[bottom]=mNo[top];
            mNo[top]=temp;
        }
        else{
            loop=0;
            cutPoint=top;
                }
    }

    return cutPoint;
}

//recusive function that will partition the sublist until there is no sublist left
void quickSort(int mNo[], int first, int last){
    int cut;
    if(first<last){
        cut=partition(mNo,first,last);
        quickSort(mNo,first,cut);
        quickSort(mNo,cut+1,last);
    }
}

//function to display selection menu
int optionchoose(){
    int choice;

    for(int i=0;i<66;i++){
        cout<<"_";
```

```
    }
    cout<<endl<<endl;

    cout<<"Select the following options\n"
        <<"1. Student Registration (STUDENT ONLY)\n"
        <<"2. College Room Status [Includes room booking and payment] (STUDENT ONLY)\n"
        <<"3. View Student's booked room. (STAFF ONLY)\n"
        <<"4. Search Student's Details (BOTH CAN ACCESS THIS SECTION)\n"
        <<"5. Manage college room. (STAFF ONLY)\n"
        <<"6. View and delete student personal details. (STAFF ONLY)"<<endl;
    cout<<"-----------------------------------------------------------------\n";
    cout<<"Your choice : ";
    cin>>choice;

    return choice;
}

//class to save login info and verified user
class Login{
    private: //private class functions
        string username;
        int password;
        char identification;

    public: //public class functions
        Login(){
            username = " ";
            password = 0;
            identification = ' ';
        }

        void getDetails(string u, int p,char i){
            username = u;
            password = p;
            identification = i;
        }

        void SetUser(char i){
            identification = i;

            if(toupper(identification) == 'U'){
                cout << "\nYou are verified as user login."<<endl;
            }
            else if(toupper(identification) == 'A'){
                cout << "\nYou are verified as admin login."<<endl;
            }
        }

        bool FillCredentials(int p){
            if(p == password)
                return true;
            else
                return false;
        };
};

//class that handles student registration for college
class RegistrationCollege{
    private: //private class functions
        string fname, lname, fullname;
        string course,college;
        char gender;
        int matricNum, digit, tempmatric; //validation conducted based on tempmatric

    public: //public class functions
        RegistrationCollege(){
            fullname=" ";
            fname=" ";
            lname=" ";
            tempmatric = 0;
            matricNum = 0;
            gender = ' ';
            course = " ";
            college = " ";
        }
```

```
        void studRegister(string &fname,int &matNum,string &cou,string &tempcollege, int
&sizearray){ //get student data
            cout<<"WELCOME TO STUDENT REGISTRATION FOR COLLEGE SECTION !"<<endl;
            cout<<"-----------------------------------------------------------------
"<<endl<<endl;

            cout<<"Full name (FIRST NAME AND LAST NAME ONLY example: John Doe) : ";
            getline(cin,fname);
            getline(cin,lname);
            fullname= fname + " " + lname;
            fname=fullname;

            cout<<"Matric No (PLEASE STATE LAST FOUR DIGITS ONLY)              : ";
            cin>>matricNum;

            tempmatric = matricNum;

            digit = 0;

            while (tempmatric != 0){
                tempmatric = tempmatric/10;
                digit++;
            }

            while(digit > 4 || digit < 4){
                cout<<"\nInvalid number of digits for matric number. Please re-enter
again. "<<endl<<endl;

                cout<<"Matric No (PLEASE STATE LAST FOUR DIGITS ONLY)              : ";
                cin>>matricNum;

                tempmatric = matricNum;

                digit = 0;

                while (tempmatric != 0){
                    tempmatric = tempmatric/10;
                    digit++;
                }
            }

            matNum = matricNum;

            cout<<"Gender (M = male, F = female)                               : ";
            cin>>gender;

            while(gender != 'M' && gender != 'F'){
                cout<<"\nInvalid gender code. Please re-enter again. "<<endl<<endl;

                cout<<"Gender (M = male, F = female)                               : ";
                cin>>gender;
            }

            cout<<"Course (Please state in the form of SECJ)                   : ";
            cin>>course;
            cou=course;

            cout<<"College desired (KTDI,KTF,KSI,KYC and KFG only)             : ";
            cin>>college;
            tempcollege = college;

            sizearray++;
            cout<<endl;
            cout<<sizearray<<" student(s) registered."<<endl;

        }

        void RegStudOutput(){ //display student data
            cout<<"Full name  :"<<fullname<<endl;
            cout<<"Matric No. : "<<matricNum<<endl;
            cout<<"Gender     : "<<gender<<endl;
            cout<<"Course     : "<<course<<endl;

        }
};

//class that handles college room and assign it to the student registered
```

```
class CollegeRoom{
    private: //private class functions
        int roomNo,floor,available[collegelist],occupied[collegelist];
        char wingBuild;
        string collegeName[collegelist], roomhold;

    public: //public class functions

        //default constructor
        CollegeRoom(){
            roomNo = 0;
            floor = 0;
            wingBuild = ' ';
            roomhold=" ";
        }

        //implement basic sequential search for the college
        string setCollegeBuilding(string c){
            string searchColl, emptycoll = " "; //means college name wanted is not
available

            for(int temp = 0; temp < 5; temp++){
                if(c == collegeName[temp]){
                    searchColl = c;
                    cout<<"\nYour desired college is available here."<<endl;
                    available[temp] = available[temp]-1;
                    occupied[temp] = occupied[temp]+1;
                    return searchColl;
                    break;
                }
            }

            cout<<"\nSorry, the college you input is not in the system. Please fill in the
registration form again."<<endl;

            return emptycoll;
        }

        //read college list from the data input
        void readCollege(){
            collroom.open("collegelist.txt",ios::in);

            if(!collroom){
                cout<<"Input file is not found !!"<<endl;
                exit (0);
            }

            for(int i=0; i<5;i++){
                collroom >> collegeName[i] >> available[i] >> occupied[i];
            }

            collroom.close();
        }

        //display the college data
        void displayCollege(){
            cout<<"\nLIST OF COLLEGE AVAILABLE"<<endl;
            cout<<"------------------------------------------------------------------
"<<endl;

            cout<<"College Name\t"
                <<"Available rooms \t"
                <<"Room occupied\t"<<endl;
            cout<<"------------------------------------------------------------------
"<<endl;

            for(int i=0; i<5; i++){
                cout << collegeName[i]<<"     \t\t"
                     << available[i]<<"\t\t\t"
                     << occupied[i] << endl<<endl;
            }
        }

        //assign room number, floor and wing building
        void assignroom(string &room){
            string result;
```

```
            cout<<"\nROOM REGISTRATION"<<endl;
            cout<<"------------------------------------------------------------------
"<<endl;

            do{
                cout<<"Please choose which room number you prefer (1-30 only): ";
                cin>>roomNo;

                if(roomNo<1 || roomNo>30)
                    cout<<"\nSorry, there are only 30 rooms available for each college.";
            } while (roomNo<1 || roomNo>30);

            if(roomNo>=1 && roomNo<=10){
                floor = 0;
                if(roomNo>=1 && roomNo<=5){
                    wingBuild = 'A'; //left wing
                }else if(roomNo>=6 && roomNo<=10){
                    wingBuild = 'B'; //right wing
                }
            }else if(roomNo>=11 && roomNo<=20){
                floor = 1;
                if(roomNo>=11 && roomNo<=15){
                    wingBuild = 'A'; //left wing
                }else if(roomNo>=16 && roomNo<=20){
                    wingBuild = 'B'; //right wing
                }
            }else if(roomNo>=21 && roomNo<=30){
                floor = 2;
                if(roomNo>=21 && roomNo<=25){
                    wingBuild = 'A'; //left wing
                }else if(roomNo>=26 && roomNo<=30){
                    wingBuild = 'B'; //right wing
                }
            }
            room = to_string(roomNo) + "-" + wingBuild + to_string(floor);
            roomhold=room;

            cout<<"Your room no is "<<room<<endl;

            cout<<"Note that the number in front is the floor level."<<endl;

            if(wingBuild == 'A')
                cout<<"A = your room is on the left side of the building."<<endl;
            else if(wingBuild == 'B'){
                cout<<"B = your room is on the right side of the building."<<endl;
            }
        } //assign the room number

        void displayRoom(){
            cout<<"Room No    : "<< roomhold <<endl;
        }

        string returnroom(){
            return roomhold;
        }
};

//handles check in, check out and duration
class CheckInOut{
    private: //private member functions
    string tempin, tempout;
    int duration, tempdur;

    public: //public member functions

    //default constructor
    CheckInOut(){
        tempin = " ";
        tempout = " ";
        duration = 0;
    }

    //return duration value
    int returnduration(){
        return tempdur;
    }
```

```
    //get input from user to fill in details for check in, check out and duration
    void studCheckInOut(string &indate, string &outdate, int &duration){ //collect data
        cout<<"\nSTUDENT INFORMATION FOR CHECK IN AND CHECK OUT"<<endl;
        cout<<"-----------------------------------------------------------------"<<endl;

        cout<<"Check in date (Please put in the form of DD/MM/YYYY)  : ";
        cin>>indate;
        tempin=indate;

        cout<<"Check out date (Please put in the form of DD/MM/YYYY) : ";
        cin>>outdate;
        tempout=outdate;

        cout<<"Duration of stay (Days) : ";
        cin>>duration;
        tempdur=duration;
        cout<<"-----------------------------------------------------------------"<<endl;
    }

    //display check in, check out and duration
    void displayCheckInOut(){
        cout<<"-----------------------------------------------------------------"<<endl;
        cout<<"Check in date      : "<<tempin<<endl;
        cout<<"Check out date     : "<<tempout<<endl;
        cout<<"Duration of stay   : "<<tempdur<< " days."<<endl;
    }

};

//global object declaration
CheckInOut chkinout;

//handles payment
class Payment{
    private: //private member functions
        float finalpayment, temppay;
        int rate=2; //rate per day

    public: //public member functions

        //default constructor
        Payment(){
            finalpayment=0.00;
        }

        //calculate payment
        void calcpay(float &finalpayment){
            finalpayment=chkinout.returnduration()*rate;
            temppay=finalpayment;
        }

        //display final payment
        void displayPayment(){
            cout<<"\nPrice rate per day : RM 2"<<endl;
            cout<<"Please pay your final amount RM "<<setprecision(2)<<fixed<<temppay<<"
at your block office"<<endl;
        }
};

//global object declarations
Login temp;
RegistrationCollege regcol;
CollegeRoom colreg;
Payment pay;

//handles receipt display
class Receipt{
    public: //public class functions

    //display receipt
    void displayPayment(){

        cout<<"AUTOMATED RECEIPT"<<endl;
        cout<<"-----------------------------------------------------------------"<<endl;
        regcol.RegStudOutput();
        colreg.displayRoom();
        chkinout.displayCheckInOut();
```

```
        pay.displayPayment();
        cout<<"--------------------------------------------------------------------"<<endl;
        cout<<"THANK YOU !! ENJOY YOUR STAY <3"<<endl;
    }
};

class StaffView{
    private: //private class functions
    fstream sv;

    public: //public class functions

    //display all inputs of the users
    void StaffViewOutput(string fullname[], int matric[], string course[], string
college[],string room[], string in[], string out[], int duration[], float pay[], int
sizearray){
        char decide;
        int first = 0, last = sizearray;

        sv.open("studentregistered.txt",ios::out);

        if(!sv){
            sv <<"Data output file is not available."<<endl;
            exit(0);
        }

        sv << "STUDENT'S REGISTERED"<< endl;

        for(int i=0;i<250;i++){
            sv<<"_";
        }
        sv<<endl;

        sv <<setw(5)<<left<<"No."
           <<setw(20)<<"Full name"
           <<setw(12)<<"Matric"
           <<setw(8)<<"Course"
           <<setw(20)<<"College Registered"
           <<setw(10)<<"Room No"
           <<setw(15)<<"Check In Date"
           <<setw(18)<<"Check Out Date"
           <<setw(10)<<"Duration"
           <<setw(15)<<"Payment required (RM)"<< endl;

        for(int i=0;i<250;i++){
            sv<<"_";
        }
        sv<<endl;

       for(int i=0;i<sizearray;i++){
            sv <<setw(5)<<left<< (i+1)
               <<setw(20)<<fullname[i]
               <<setw(12)<<matric[i]
               <<setw(8)<<course[i]
               <<setw(20)<<college[i]
               <<setw(10)<<room[i]
               <<setw(15)<<in[i]
               <<setw(18)<<out[i]
               <<setw(10)<<duration[i]
               <<setw(15)<<setprecision(2)<<fixed<<pay[i]<<endl;
        }

        for(int i=0;i<250;i++){
            sv<<"_";
        }
        sv<<endl;

        cout<<"Do you want to sort the data based on ASCENDING matric no(Y/N)?"<<endl;
        cout<<"Your choice : ";
        cin>>decide;

        if(toupper(decide) == 'Y'){
            if(sizearray>=3){
                //display sorted array with ascending order
                quickSort(matric, first, last);
                sv <<"\n--------------------------------------------------------------------
"<<endl;
```

```
                sv <<"Sorted matric no : ";
                for(int i=0;i<sizearray;i++){
                    sv<<matric[i]<<" ";
                }
                sv <<"\n------------------------------------------------------------------------
"<<endl;
            }
            else
                cout<<"Sorry! There is not enough elements to be sorted"<<endl;
        }else if(toupper(decide) == 'N'){
            //display original array with no sorting
            sv <<"\n------------------------------------------------------------------------
"<<endl;
            sv<<"Unsorted matric no: ";
            for(int i=0;i<sizearray;i++){
                sv<<matric[i]<<" ";
            }
            sv <<"\n------------------------------------------------------------------------
"<<endl;
        }

        sv.close();
    }
};

//class Node to store data from the system
class Node{
    public:
        int data;
        Node *next;
};

//perform insertion and searching nodes
class List{
    private:
        Node *root;
        Node *head;

    public: //public class functions

        //constructor
        List(void){
            root = NULL;
        }

        //destructor
        ~List(void){
            Node *currNode = root, *nextNode = NULL;

            while(currNode != NULL){
                nextNode = currNode->next;

                delete currNode;
                currNode = nextNode;
            }
        }

        //determine if the head is empty
        bool IsEmpty(){
            if (root->next == head && root->data==0)
                return true;
            else
                return false;
        }

        void InsertNode(Node **head, int item){
            Node *temp = new Node;
            Node *ptr;

            temp->data=item;
            temp->next=NULL;

            if(*head == NULL){
                *head = temp;
            }else{
                ptr= *head;
                while(ptr->next != NULL)
```

```
                ptr = ptr->next;
            ptr->next = temp;
        }
    }

    //transfer element from the array to the nodes
    Node *arrayToList(int matric[],int sizearray){

        for (int i = 0; i < sizearray; i++){
            InsertNode(&root, matric[i]);
        }

        //check if the list is empty or not
        if(IsEmpty()){
            cout<<"\nThe list is empty!\n"<<endl;
        }else{
            cout<<"\nThe list is not empty!\n"<<endl;
        }

        return root;
    }

    //search if the matric searched is available in the data
    bool FindNode(int matricchoose){
        Node *currNode = root;

        while(currNode != NULL && currNode->data == matricchoose){
            currNode = currNode->next;
        }

        if(currNode == NULL){
            cout<<"\nMatric Number search is available !!"<<endl;
            return true;
        }else{
            cout<<"Sorry. The matric number searched is not available !!"<<endl;
            return false;
        }

    }
};

//class to search the student details
class StaffSearch{
    public: //public class functions

        //function to search student and display student details
        void SearchStudent(string fullname[], int matric[], string course[], string
college[],string room[], string in[], string out[], int duration[], float pay[], int
sizearray){
            int code, correct;
            List *temp = new List;
            Node *root = temp -> arrayToList(matric, sizearray);

            cout<< "Please enter the Student Matric Number: ";
            cin >> code;

            if(temp->FindNode(code)){

                for(int i=0;i<sizearray;i++){

                    if (matric[i]==code){
                        cout<<"\nThe following student's matric number is available
!!"<<endl;

                        cout<<"Full Name : " <<fullname[i]<<endl;
                        cout<<"Matric No : " <<matric[i]<<endl;
                        cout<<"Course    : " <<course[i]<<endl;
                        cout<<"College   : " <<college[i]<<endl;
                        cout<<"Room      : " <<room[i]<<endl;
                        cout<<"Check In  : " <<in[i]<<endl;
                        cout<<"Check Out : " <<out[i]<<endl;
                        cout<<"Duration  : " <<duration[i]<<" days"<<endl;
                        cout<<"Payment   : RM " <<pay[i]<<endl;
                    }
                }
            }else{
```

```
                cout<<"There are no data of the following student details in the
system."<<endl;
            }

        }
};

//class that display room no based on the latest day and date
class RoomStack{

    private:
        int top=-1;
        string roomdata[10],indata[10], outdata[10];
    public:

        int isFull(){
            if(top==10-1)
                return 1;
            else
                return 0;
        }

        int isEmpty(){
            if(top==-1)
                return 1;
            else
                return 0;
        }

        void stackpush(string roomNo[], string chkin[], string chkout[], int sizearray){

            cout<<"Transfering data. Please wait a moment...."<<endl;
            sleep(2); //delay

            if(isFull()){
                cout<<"Stack is full."<<endl<<endl;
            }
            else{
                for(int i=0; i<sizearray;i++){
                    top = top+1;
                    roomdata[i]=roomNo[i];
                    indata[i] = chkin[i];
                    outdata[i]=chkout[i];
                }

                cout<<"\nData for roomNo, check in and checkout has been inserted."<<endl;
            }
        }

        //remove room no
        void stackpop(){
            string temproom, tempin, tempout;

            if(isEmpty()){
                cout<<"Stack is empty."<<endl;
            }
            else{
                cout<<"Room no - ("<< roomdata[top] << ") and its check in and check out
date has been removed."<<endl;

                temproom = roomdata[top];
                tempin = indata[top];
                tempout = outdata[top];
                top=top-1;
            }
        }

        //display stack elements based on LIFO
        void stackdisplay(){
            if (top>=0){
                cout<<"\nROOM BOOKING WITH CHECK IN AND CHECK OUT DATES"<<endl;
                cout<<"------------------------------------------------"<<endl<<endl;
                cout<<setw(15)<<left<<"Room Number"
                    <<setw(20)<<"Check In Date"
                    <<setw(20)<<"Check Out Date"<<endl;

                for (int i=top; i>=0; i--)
```

```
                    cout<<setw(15)<<left<<roomdata[i]
                        <<setw(21)<<indata[i]
                        <<setw(19)<<outdata[i]<<endl;
            } else
                cout<<"Stack is empty"<<endl;
        }
};

//struct to store pointer for queue implementation
struct QNode{

    public:
        char cod[5];
        int mat;
        struct QNode *next1;
};

//class to view and delete student personal details
class MatricQueue{
    public:

        //point initializer
        struct QNode *front1 = NULL;
        struct QNode *rear1 = NULL;
        struct QNode *temp1;

        int val = 0;
        char code[5]; //code stands for course

        //function to insert element from array into nodes
        void QueueInsert(int matric[],string course[],int sizearray){

            cout<<"Transfering data. Please wait a moment...."<<endl;
            sleep(2); //delay

            for(int i=0; i<sizearray; i++){
                val = matric[i];

                //convert string to char
                int n = course[i].length();

                strcpy(code, course[i].c_str());

                if (rear1 == NULL){
                    rear1 = (struct QNode *)malloc(sizeof(struct QNode));
                    rear1->next1 = NULL;
                    rear1->mat = val;

                    //input char into struct cod
                    for(int i=0; i<n;i++){
                        rear1->cod[i]=code[i];
                    }

                    front1 = rear1;
                }else{
                    temp1 = (struct QNode *)malloc(sizeof(struct QNode));
                    rear1->next1 = temp1;
                    temp1->mat = val;

                    //input char into struct cod
                    for(int i=0; i<n;i++){
                        temp1->cod[i]=code[i];
                    }

                    temp1->next1 = NULL;
                    rear1 = temp1;
                }

                cout<<"All data of student details have been assigned to the
nodes."<<endl;
            }
        }

        //function to delete element at the top based on queue concept
        void QueueDelete(){
            temp1=front1;
```

```
            if (val == 0){
                cout<<"Queue is empty"<<endl;
            }
            else if(front1==NULL)
            {
                cout<<"Underflow"<<endl;
                return;
            }
            else if (temp1->next1!=NULL)
            {
                temp1=temp1->next1;
                cout<<"Student details deleted from queue with matric number : "<<front1-
>mat;
                cout<<endl;

                free(front1);
                front1=temp1;
            }
            else
            {
                cout<<"Student details deleted from queue with matric number : "<<front1-
>mat;
                cout<<endl;

                free(front1);
                front1=NULL;
                rear1=NULL;
            }

        }

        //function to display the queue element based on FIFO
        void QueueDisplay(){
            temp1=front1;
            if ((front1==NULL) && (rear1==NULL) || val == 0)
            {
                cout<<"Queue is empty"<<endl;
                return;
            }

            cout<<"STUDENT PERSONAL DETAILS"<<endl;
            cout<<"-------------------------------------------------"<<endl<<endl;
            cout<<setw(20)<<left<<"Matric number   "
                <<setw(20)<<"Course code"<<endl;

            while (temp1!=NULL)
            {
                cout<<setw(20)<<left<<temp1->mat;

                for(int j=0;j<4;j++){
                    cout<<temp1->cod[j];
                }

                cout<<endl;

                temp1=temp1->next1;
            }
            cout<<endl;
        }
};

//global object declarations
Receipt receipt;
StaffView staff;
StaffSearch search;
RoomStack stack;
MatricQueue queue;

//main function
int main(){

    int mNum, sizearray=0,option, password;
    float paid;
    char userType, confirmation;
    string username,fullname,course, tempcollege, tempsearch = " ", room;
    string in,out;
    bool use, credentials;
```

```
    string fullnamearray[5]={" "," "," "," "," "},coursearray[5]={" "," "," "," "," "};
    string collegearray[5]={" "," "," "," "," "}, roomarray[5]={" "," "," "," "," "};
    string cinarray[5]={" "," "," "," "," "}, coutarray[5]={" "," "," "," "," "};
    int matricarray[5]={0,0,0,0,0},durationarray[5]={0,0,0,0,0};
    float payarray[5]={0,0,0,0,0};
    int i=0, k=0, duration;

    colreg.readCollege(); //read data input file collegelist.txt

    do{
        system("CLS");

        cout<<"WELCOME TO G8 MINI PROJECT HOSTEL MANAGEMENT SYSTEM !!"<<endl;

        do{
            cout<<"-----------------------------------------------------------------\n"<<endl;
            cout<<"Select which user? U == student/guest, A == admin"<<endl;
            cout<<"Your choice : ";
            cin>>userType;

            if(toupper(userType) == 'U'){
                temp.SetUser(toupper(userType));
            }
            else if(toupper(userType) == 'A'){
                temp.SetUser(toupper(userType));
            }
            else
                cout<<"\nYou have entered the wrong input. Please try again."<<endl;

        }while(toupper(userType) != 'U' && toupper(userType) != 'A');

        cout<<"\nPlease enter your username and password"<<endl;

        cout<<"Username : ";
        cin>>username;

        cout<<"Password : ";
        cin>>password; //need to set if match with the system

        temp.getDetails(username,password,userType);
        //send.Staff(username,password,usertype);

        do{
            cout<<"Please confirm again your password : ";
            cin>>password;

            credentials = temp.FillCredentials(password);

            if(credentials == 0){
                cout<<"\nPassword not match with the system. Please input again"<<endl;
            }else if(credentials == 1){
                cout<<"\nSuccessful login!"<<endl;
            }

        }while(credentials == 0);

        option = optionchoose();

        switch(option){
            case 1:
                if(toupper(userType) == 'U')
                {
                    system ("cls");
                    cout<<endl;
                    regcol.studRegister(fullname, mNum, course, tempcollege, sizearray);

                     if(fullnamearray[i]==" " && coursearray[i]== " " &&
collegearray[i]==" " && matricarray[i]==0){
                        fullnamearray[i]=fullname;
                        matricarray[i]=mNum;
                        coursearray[i]=course;
                        collegearray[i]=tempcollege;
                        i++;
                    }

                    while (sizearray==0){
```

```
                        cout<<"\nNo students registered. Please do the registration first
before booking."<<endl;
                        break;
                    }

                    cout<<"\n"<<"Press ENTER to continue...";
                    cin.ignore();
                    cin.get();

                    cout<<flush;

                }else{
                    cout<<"\nSorry. Only student login can access this section."<<endl;
                    break;
                }

            case 2:
                if(toupper(userType) == 'U')
                {
                    system("CLS");
                    colreg.displayCollege();
                    tempsearch = colreg.setCollegeBuilding(tempcollege);

                    cout<<"\n"<<"Press ENTER to continue...";
                    cin.ignore();
                    cin.get();

                    cout<<flush;
                    system("CLS");

                    if(tempsearch==" "){ //user need to register again since their college
input is wrong.
                        break;
                    }

                    colreg.assignroom(room);
                    cout<<"\n"<<"Press ENTER to continue...";
                    cin.ignore();
                    cin.get();

                    cout<<flush;
                    system("CLS");

                    chkinout.studCheckInOut(in,out,duration);
                    cout<<"\n"<<"Press ENTER to continue...";
                    cin.ignore();
                    cin.get();

                    cout<<flush;
                    system("CLS");

                    pay.calcpay(paid);
                    receipt.displayPayment();

                    if(roomarray[k]== " " && cinarray[k]==" " && coutarray[k]==" " &&
durationarray[k]==0 && payarray[k]==0){
                        roomarray[k] = room;
                        cinarray[k] = in;
                        coutarray[k] = out;
                        durationarray[k] = duration;
                        payarray[k] = paid;

                        k++;
                    }

                }else{
                    cout<<"Sorry. Only student login can access this section."<<endl;
                }
                break;

            case 3:
                if(toupper(userType) == 'A'){
                    system("CLS");
                    staff.StaffViewOutput(fullnamearray, matricarray, coursearray,
collegearray, roomarray, cinarray, coutarray, durationarray, payarray, sizearray);
                }else
```

```
                    cout<<"Sorry. You are not login as the staff. Please relogin again to
access this section."<<endl;
                break;

            case 4:
                search.SearchStudent(fullnamearray, matricarray, coursearray,
collegearray, roomarray, cinarray, coutarray, durationarray, payarray, sizearray);
                break;

            case 5:
                if(toupper(userType) == 'A'){

                    system("cls");

                    stack.stackpush(roomarray, cinarray, coutarray, sizearray);

                    do{
                        system ("cls");
                        cout<<"VIEW AND DELETE STUDENT ROOM DETAILS"<<endl;
                        cout<<"-------------------------------------------------------
"<<endl;
                        cout<<"\nOptions : \n"
                            <<"1. Delete past room details. [POP]\n"
                            <<"2. Display room details. [DISPLAY]\n"
                            <<"3. Exit"<<endl;
                        cout<<"Your choice: ";
                        cin>>option;
                        cout<<endl;

                        if(option == 1){
                            stack.stackpop();

                            cout<<"\n"<<"Press ENTER to continue...";
                            cin.ignore();
                            cin.get();

                            cout<<flush;

                        }else if(option == 2){
                            stack.stackdisplay();

                            cout<<"\n"<<"Press ENTER to continue...";
                            cin.ignore();
                            cin.get();

                            cout<<flush;
                        }else{
                            cout<<"You are about to return to the main menu."<<endl;
                        }

                    }while(option == 1 || option == 2);

                }
                break;

            case 6:
                if(toupper(userType) == 'A'){

                    system("cls");
                    queue.QueueInsert(matricarray,coursearray,sizearray); //include name
and course code

                    do{
                        system("cls");
                        cout<<"VIEW AND DELETE STUDENT PERSONAL DETAILS"<<endl;
                        cout<<"-------------------------------------------------------
"<<endl;
                        cout<<"\nOptions : \n"
                            <<"1. Delete past student personal details from top. [POP]\n"
                            <<"2. Display remaining students details. [DISPLAY]\n"
                            <<"3. Exit"<<endl;
                        cout<<"Your choice: ";
                        cin>>option;
                        cout<<endl;

                        if(option == 1){
                            queue.QueueDelete();
```

```
                            cout<<"\n"<<"Press ENTER to continue...";
                            cin.ignore();
                            cin.get();

                            cout<<flush;

                        }else if(option == 2){
                            queue.QueueDisplay();

                            cout<<"\n"<<"Press ENTER to continue...";
                            cin.ignore();
                            cin.get();

                            cout<<flush;

                        }else{
                            cout<<"You are about to return to the main menu."<<endl;
                        }

                    }while(option == 1 || option == 2);
                }

                break;

        }

        cout<<"\nDo you want to relogin? (Y = yes / N = no)"<<endl;
        cout<<"Your choice : ";
        cin>>confirmation;

        system("cls");

    }while(toupper(confirmation) == 'Y');

    return 0;

}
```