Uhrzeit	Montag	Dienstag	Mittwoch	Donnerstag	Freitag
7:55	Deutsch	Französisch	Biologie	Religion	Mathe
8:40	Englisch	Musik	Geschichte	Physik	Mathe
9:40	Sport	Geschichte	Mathe	Französisch	Englisch
10:25	Sport	Mathe	Englisch	Französisch	Chemie
11:25	Religion	Physik	Erdkunde	Erdkunde	Musik
12:10	Sozialkunde	Chemie	Deutsch	Deutsch	Bio

TIME TABLE GENERATOR

Using Python

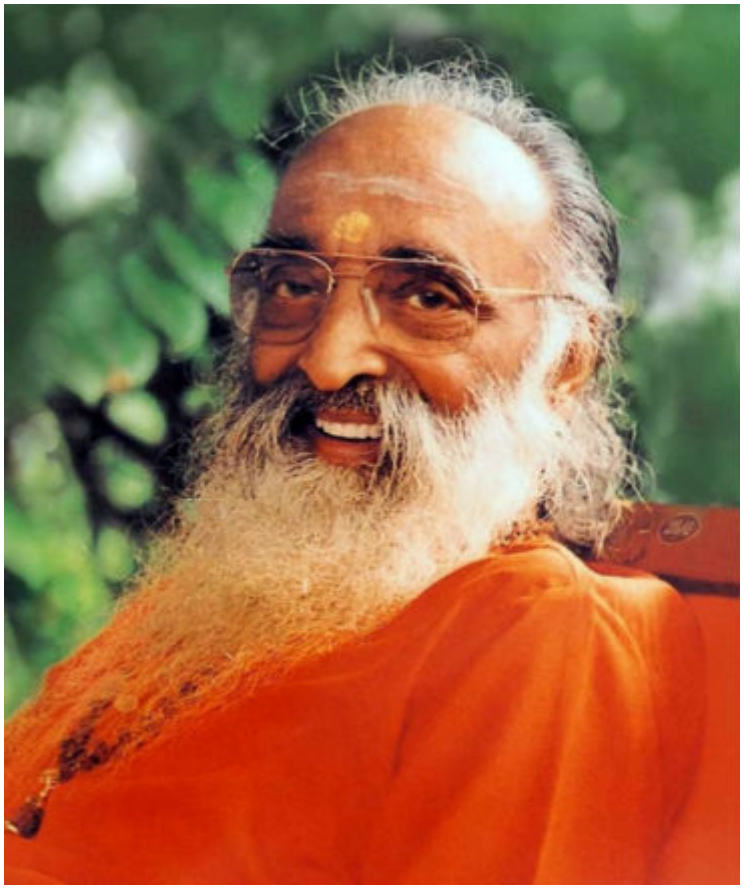
ABSTRACT

Implementation of a time table generator in Python using basics of objected oriented programming and simple algorithms.

Tarun G and Netra Gupta

Computer Science

CHINMAYA VIDYALAYA



Term 2

Computer Science Project

Time Table Generator

Tarun G - .....

Netra Gupta - .....

CHINMAYA VIDYALAYA
CHENNAI

Register No. ……

Certified to be the Bonafide Record of work done by

……

Of Class …… in the CHINMAYA VIDYALAYA, CHENNAI.

During the Year ……

Dated …… Teacher-in charge

Submitted for Practical Examination held on ……

at ……

Chennai

Dated ……

Principal External Examiner

ACKNOWLEDGEMENT

We wish to express our deep gratitude and sincere thanks to the Principal, Chinmaya Vidyalaya, Smt. Usha Nandhini for her encouragement and facilities she provided for this project work, without which the project would have been a distant reality.

We extend our hearty thanks to Mr. Sekar K, who guided us for the successful completion of this project. We take this opportunity to express our deep sense of gratitude for his invaluable guidance, constant encouragement, constructive comments, sympathetic attitude, and immense motivation which has sustained our efforts at all stages of this project.

We also thank our institution and our faculty members for being there as a helping hand and guidance.

We also extend our heartfelt gratitude to our family and relatives, who have been constant pillars of support and guidance throughout the completion of the project.

INDEX

1. Synopsis

Title of the Project: School Timetable Generator

Definition: Design a project to generate time tables for our school for the entire academic year. Display data in visually pleasing format with schedules individually assigned to all classes and all teachers of the school, as well as information about substitution teachers.

Contribution / Team members: Netra Gupta and Tarun G

Detail: The Project "Time Table Generator" is developed jointly by Netra Gupta and Tarun G. It took approx. 2 months to develop this project, working 8 hours weekly. The project was entirely made by us.

Objective:

- Generate a school timetable for all classes of our school, ensuring adequate distribution of subjects for the classes and reasonable schedules for the teachers.
- Reduce time wastage in doing work manually which can be efficiently automated.

Minimum Hardware Requirements:

Operating System-Windows 7 or above

x86 64-bit CPU (Intel / AMD architecture)

2 GB RAM. 10 MB free disk space.

Python 3.7.x or higher

Note: Newer processors are better. Program may run slower on older processors.

Limitations:

1. It is not a web based project, therefore the command line might appear confusing to non-tech-savvy people.
2. Needs modifications to fulfill the needs of every school.
3. Code functionality and features are limited.
4. Final data set cannot be accessed with commands and has to be done manually

2. Why this Project?

This project aims to:

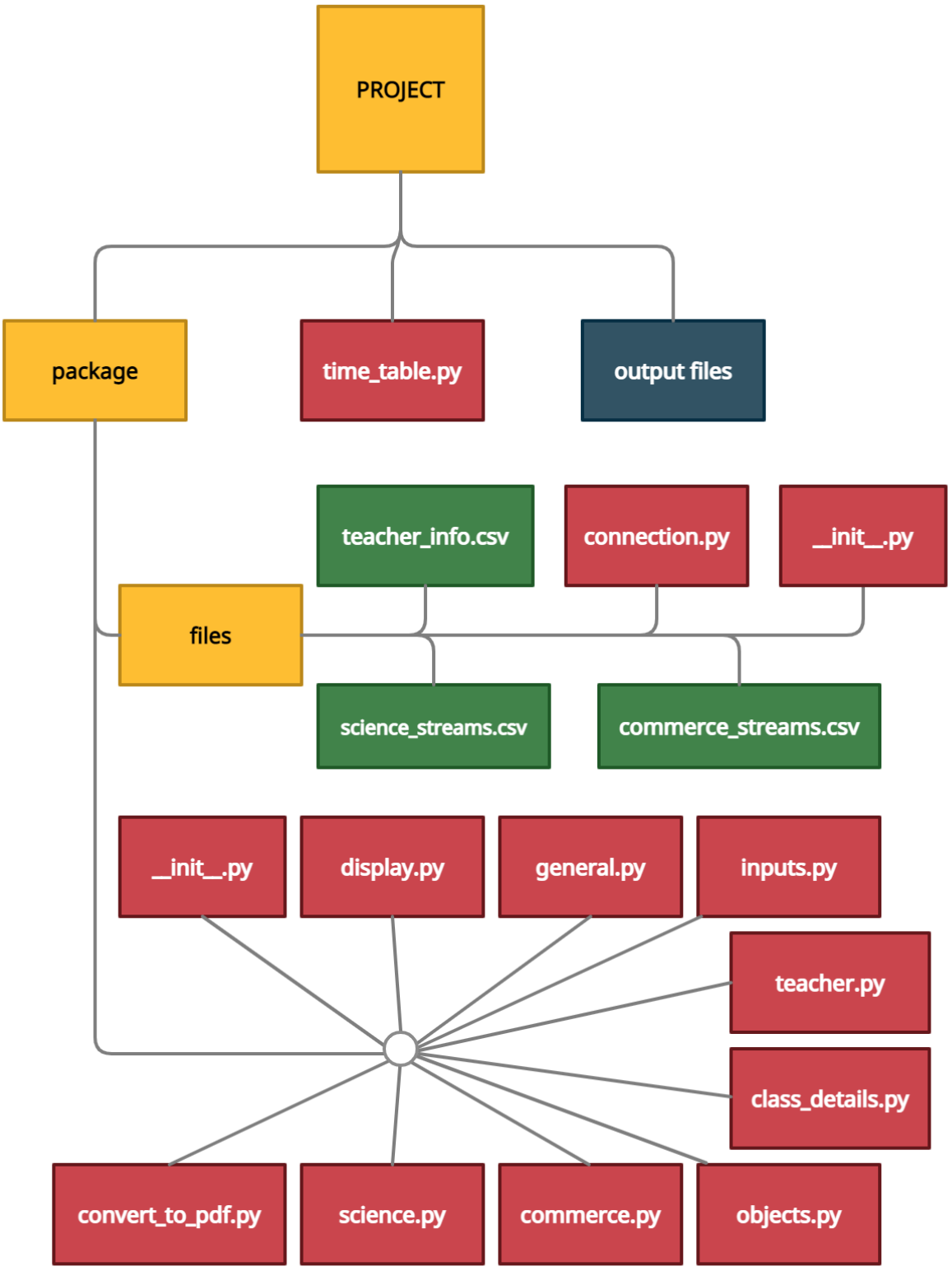
- Generate a school timetable for all classes of our school.
- Reduce time wastage in doing work manually which can be efficiently automated.

We found that roughly about a week or so is spent by teachers every year in creating a suitable timetable which satisfies all necessary requirements, making it quite a tedious job as there are plenty of criteria which have to be met. Therefore, to reduce the extra workload, we decided to make a timetable generator which can satisfy all requirements and can also be modified, if necessary, according to future needs of the school.

3. Pre-Requisite Information

- Basic knowledge on usage of python such as if-elif-else statements, while and for loops etc.
- Creating, managing and modifying data types like lists, tuples and dictionaries and performing basic operations on them.
- Creating and calling functions using def keyword, return statements
- Basics of object oriented programming (OOP), creating classes.
- Accessing files, reading/writing data and managing them.
- Importing modules and directories like os, system, random, csv and using them
- Creating modules and importing functions from them.

4. Directory Structure



5. Working of the Project

The main python project file *time_table.py* is the backbone of the project. It contains multiple important functions that aim to assign classes to teachers as per requirements of our school while making sure that no two classes overlap for a particular teacher. The code first reads a file containing the list of all teachers and their details such as the subject and classes they teach and stores them in a list. The code also contains details of the classes i.e. which classes have which subjects. Then it proceeds to assign the subject teachers for the respective classes using a user defined function called *min_max()*. *min_max()* ensures the even distribution of teachers for all classes they teach while making sure that they do not get excess classes. It then goes on to generate the time tables and displays the output in a text file.

We also used the concept of modules and packages to make the code look neater. We needed to define some functions to display various information. These functions are stored in a separate file *display.py*. The functions of this file were then imported to the main file *time_table.py* and then used. Further, when we take the input from the user, it is important that valid details are entered. To check these details, we created another file *inputs.py*, whose functions ensure that all details entered by the user are acceptable and can be handled by the code without any errors.

To handle the large amounts of data within the python program, we created two objects, one for a classroom and one for a teacher. The classroom object is named as Grade, teacher object is named as Teacher. These are stored in the file *objects.py*. Grade object has various attributes like schedule, main subjects, additional subjects, and its faculty. The Teacher object has attributes such as teacher ID, first name, last name, assigned classes, max classes, min classes and their schedule.

5.1 Overall Logic

1. Read all the teacher data from *teacher_info.csv* (file storing all teacher details)
2. Define all the data for all the classes (within the program)
3. Display home screen, ask user what they want to do
4. If choice is to edit teacher details, go to step 7
5. If choice is to generate time tables, go to step 10
6. If choice is exit, the program ends
7. Ask user whether they want to add, remove or update a teacher
8. Perform required operation requested by user
9. Go to step 3
10. Define all the classes using data from step 2 and store in a list
11. Iterate through all the classes
12. Assign all the required faculty for each class
13. If two classes have certain common subjects, assign faculty commonly
14. Once all faculty is assigned, start generating time tables
15. Iterate through all the classes, call the generation functions
16. If one class doesn't need to by synced with another class, call *generate()*
17. If two classes need some subjects synced, then call *generate_together()*
18. Once all time tables are generated successfully, user inputs a filename for output
19. Save the output file in .txt and .pdf formats
20. Exit

5.2 Data Handling

The file teacher.py contains all necessary functions for handling the data stored in *teacher_info.csv*. *teacher_info.csv* is the main data file of the program and contains all the important details of all the teachers of the school. It stores information in the format

Teacher ID, first name, last name, subject, classes taught, min classes, max classes

The files *science.py* and *commerce.py* contain all necessary functions for handling the data stored in *science_streams.csv* and *commerce_streams.csv* respectively. In depth explanation about these two csv files are mentioned later in this document.

5.3 Assigning Teachers in a Systematic Manner

The *min_max()* function is primarily designed to assign one teacher from a list of teachers passed in as a parameter. Its main aim is to perform the task efficiently, and not just randomly. For example, two teachers teaching the same subject (suppose X) and the same grades (say 9th, 10th), then the code will equally assign classes to both teachers i.e, two from 9th and two from 10th.

The function also aims to, as the name suggests, assign teachers in a manner such that the priority is to satisfy the minimum classes requirement first, and then assign more classes once all the teachers in the list of teachers passed in as the parameter have satisfied their minimum classes requirement. For example, two teachers (teaching the same subject and for same classes, say class 9 and 10) have minimum values 1 (teacher X) and 3 (teacher Y) and maximum as 4 for both, then the code, after assigning X their first class will move on to Y and assign their minimum classes, before moving on to satisfy the class requirement and finally proceeds to satisfy the maximum value for the teachers. We made our code smart to some extent, so if the minimum and maximum classes to be taken by the entire subject department is tight, the code tries all possible ways and opts the best possible one.

Error Handling: If for whatever reason, the maximum classes of all teachers are reached and no teacher is left free, then it prints the error "maximum classes of {department name} exceeded", and then moves on to ask the user to enter more teachers.

5.4 Generating Time Tables

Once the teachers have been assigned, we need to start generating time tables. We iterate through all the classes and call the required generation functions. In some cases, for example for language and lab subjects, we need to generate time tables for two classes together because both classes would require these subjects to be in sync. While generating time tables, it is essential that the time tables of two classes or two teachers do not contradict each other. This required a lot of conditional statements to satisfy a plethora of requirements. However, the code involves a lot of random outcomes, so it is possible that it might hit a dead end and be unable to proceed further. In such cases,

we would simply restart the code and create another random generation. In some places, to reduce the randomness, we used the permutations library to check every single possibility. This part of the program is very cpu intensive, therefore we recommend more recently released processors that can perform the calculations faster.

5.5 Working of the *free_slots()* Function

The *free_slots()* function was designed to fill up the empty slots in the timetable after assigning all main and additional subjects. It was designed to assign the main teachers in a proper order. For instance, if we have 4 main subject teachers and 3 slots, the code first checks which teacher is free in which slot, then whichever slot has the least number of teachers, the code assigns that slot first and then tries to assign extra subjects evenly, thereby improving efficiency of the code! For example, let a particular class have 2 free slots. If the first free slot is given to a teacher of subject X, then it will try to assign teachers of other subjects other than X to avoid unrealistic cases where too many of the same subject is assigned to a particular class.

5.6 Classifying Science and Commerce teachers

We used the *classify_science()* and *classify_commerce()* functions to divide the senior teachers into their respective subjects and then saved that data into the files *science_streams.csv* and *commerce_streams.csv*. For example, the senior teachers who teach classes 9,10,11,12 took science as a subject for 9,10 while separately took Physics, Chemistry and Biology for practical labs and 11th and 12th! If the user entered a new teacher in commerce or science groups the code asked the user to enter the stream (phy/chem/bio for science teachers and entrepreneurship/accounts/business studies for commerce teachers). Some commerce teachers took multiple subjects so the code asked the user one by one whether the subject was taught by the teacher or not. This way we were able to avoid any error and made sure that the code ran smoothly.

5.7 Handling Language Classes

In the main code, if language is given as the subject for a particular class, we check if the section is A/B or C/D and assign Hindi, Tamil and/or Sanskrit teachers to the class and sync them as the two classes have all three subjects together(students of our school are divided into batches and can choose only one language). Section C/D had only Hindi and Tamil as their language options! The code checked the sections and languages, and made sure that both the classes and schedules of all the teachers were empty and then synced and assigned the classes.

5.8 Laboratories

Our school has 5 laboratories: Physics, Chemistry, Biology, Senior computer and junior computer labs. Whenever scheduling classes that require the use of these labs, for instance practical classes, we need to ensure that the laboratory is free and not in use. Therefore we created independent lists for all of these labs to keep track of their daily schedule as well.

For classes 9th and 10th, they have dedicated practical classes per week. For this, two sections will together be split across all three labs. Therefore it was essential to ensure that each lab is free at the same time and therefore created a unique function to check this condition.

5.9 Error Handling

We were extra cautious on how the code handles errors, so that the program does not end abruptly or raise an error. This makes our code more user friendly as even a non-python programmer can use it with ease and read through the printed message to easily fix errors!

We defined numerous functions throughout the program to ensure that every step of the program is carried out without any abrupt errors. For example, the *check_class_faculty()* function is used to ensure that there is no shortage of teachers and that a particular class has received all required teachers. In case the function identifies a shortage, the code is not allowed to continue until the user updates the existing details in the *teacher_info.csv* file.

At some places, we printed certain warning messages. For example, in the *min_max()* function, a disclaimer message is printed to suggest the user to update the existing details first, before attempting to add another teacher.

Further, since the entire program is menu driven and user friendly, when taking various inputs from the user, it is essential to ensure that only valid details are entered. Therefore we created a separate file *inputs.py* defining various functions to perform that task, whose contents are as follows:

1. *input_to_delete()* and *input_to_update()*: Ensures that the teacher ID entered by the user is an integer and is already present in the file.
2. *input_subject()* : Ensures that the subject entered by the user for a new teacher is in the list of acceptable subjects. For example, users cannot enter "social science" but must enter "sst", the term acceptable by the program.
3. *input_id() :* Ensures that the ID entered for a new teacher is an integer, unique and not already present in the file. Prints a user friendly message about which teacher is already using a particular ID, or if the ID is not acceptable.
4. *input_grades()*: Input the classes to be taught. Ensures the following:
    a. User must enter only integers from 1 to 12
    b. User must enter at least one class
    c. User cannot enter the same class twice
5. *input_min_max_classes()* : Takes input of minimum and maximum classes from the user. Ensures that
    a. Only integers are entered for both values
    b. Maximum must be greater than minimum
6. *yes_or_no():* Users must give only yes or no input using code "y" or "n". All other inputs are invalidated.
7. *input_file_name():* Input a valid file name for the output file. Users cannot include characters considered illegal by the Windows operating system. Prints a suitable message in case they do.

5.10 Substitution Teachers

If a teacher is to be absent, then the code displays a unique table of all substitute teachers that can take their classes.

For example, if it shows: 7A: 23, 25, 21

It means that the user can select any one of these free teachers and assign them to class 7A at that particular period.

We did not want to assign a substitute teacher randomly. For example, it would not make sense to assign a 5th standard teacher to a 10th standard class. Therefore we designed a priority list, where teachers who happen to teach that particular class are assigned priority. Priority is first given to teachers who already teach the class. This ensures that teaching can continue even during substitutions. In the given example, it would try to assign a math teacher or science teacher (from the class faculty) first, then consider all other teachers. This is done to ensure that learning can continue even if a teacher is to be absent.

5.11 Thinking out of the box

When we started working on this project, we were of the impression that it would be a simple task. However as more complications arose, we realized that our skill set was not broad enough. Therefore we learnt a lot of additional topics that are outside the syllabus, but were extremely useful in simplifying the code, thus making it more concise and crisp. Some of those topics are

1. Basics of object oriented programming, usage of *class* keyword
2. Saving output to text file using *sys.stdout*
3. Looping over an iterable with a counter simultaneously using *enumerate* keyword
4. Using the *isinstance* method to check whether a variable is of certain type
5. Using *all* and *any* keywords to check various conditions without multiple statements
6. *Permutations* using itertools library
7. Creating *packages* and *modules*, and importing them into the main program
8. *f-strings* for print statements, including using them for alignment and spacing
9. Python type checking using type hinting or *annotations*
10. Creating custom *exceptions* or errors to serve the purpose of the program

Note:

From python 3.7, *dictionaries are no longer an unordered collection of items*. They preserve the *insertion order* of their elements. Thus it is now legal to order dictionaries and iterate through them in whichever order we like. This was useful in one part of the program (in function *free_slots*), where we had to assign teachers based on the number of positions available, which was stored as the values in the dictionary. Therefore we needed a way to arrange the dictionary according to the values and thus we created a separate function to perform this operation using basic principles of bubble sort.

6. Collaboration in the middle of a pandemic

To make any large project with efficiency, the in-built python IDLE is not suitable. Therefore for our project we decided to use Visual Studio Code, a free code editor from Microsoft. Using an extension called Live Share, we were able to collaborate and code on the same document and communicate easily. We communicated, coded, and cross checked all the work in multiple sessions thus finishing the project ahead of schedule.

7. The Full Code

Colour Coding

The entire code is given below with specific colour coding to improve the visual appearance.

1. Purple: Keywords
2. Dark yellow: functions (both in-built and user defined)
3. Dark Red: strings and multi-line comments
4. Cyan: objects defined using class keyword (both in-built and user defined) and modules
5. Regular Blue: Keywords (def, class, True, False, None)
6. Cornflower blue: Constants (variables defined in all capitals)
7. Dark Blue: variables
8. Dark Green: In-line or single-line comments
9. Bluish-Green: Integers and floats

Contents of *connection.py*

```
"""Open a file containing data"""
import os


def open_file(name, m, new="", fil=__file__):
    """Open a file in folder 'files' from any directory on command line.
    File name and mode to open file are parameters.
    fil parameter can be passed in if file to open is in some other directory.
    Returns file object."""

    loc = os.path.realpath(os.path.join(os.getcwd(), os.path.dirname(fil)))
    file_path = os.path.join(loc, name)
    return open(file_path, m, encoding="utf-8", newline=new)
```

Contents of *display.py*

```
def display_table(t: "list[str]", scale: int = 1):
    """Print a table neatly; scale parameter controls size of table printed"""
    count = 0
    print("-"*(105*(scale) - (scale-1)*25))
    print("|", end="")
    for piece in t[1:]:
        if count == 8:
            count = 0
            print()
            print("-"*(105*(scale) - (scale-1)*25))
            print("|", end="")

```

```
            size = 10*scale
            if piece.isdigit():
                # ^10 parameter: ^ means center align, 10 ensures equal spacing
                empty = ""
                print(f" {empty:^{size}} ", end="|")
            else:
                print(f" {piece:^{size}} ", end="|")
            count += 1
        print("\n"+"-"*(92*scale - (scale-1)*22))


def home_screen() -> int:
    """Display home screen and menu and input what user wants to do"""
    print("Welcome to the Time Table Generator!")
    print("""
        1. Edit teacher details
        2. Generate time table
        3. Exit
        """)
    ch = input("Please enter your choice: ")
    valid = [str(i) for i in range(1, 4)]
    while True:
        if ch not in valid:
            print("Invalid input!")
            ch = input("Please enter your choice: ")
        else:
            ch = int(ch)
            break

    return ch


def display_teachers(data: list) -> None:
    """Displays all the information of teachers in list data"""
    lst = ["S.No.", "Name", "Subject", "Grades", "Min classes", "Max classes"]
    print(f"\n{lst[0]:>4} {lst[1]:<25} {lst[2]:<12} {lst[3]:<20} {lst[4]:^12}
{lst[5]:^12}")
    for t in data:
        full_name = t.f_name + " " + t.l_name
        grades = " ".join(str(g) for g in t.grades)
        print(f"{t.i_d:>4}. {full_name:<25} {t.subject:<12} {grades:<20}
{t.min_c:^12} {t.max_c:^12}")


def substitution_instruction():
    """Print instructions on output to inform user about substitutions"""
    print("""
    INSTRUCTIONS FOR SUBSTITUTION TEACHERS
    IF A TEACHER IS TO BE ABSENT,
    then the table shows the IDs of all available teachers.

    For example, if it shows
    7A: 23, 25, 21
    it means you can select any one of these free teachers,
    and assign them to class 7A at that particular period.

```

```
    The teachers are ORDERED from HIGHEST PRIORITY to LOWEST PRIORITY
    PRIORITY is given to teachers who already teach the class
    This ensures that teaching can continue even during substitutions
    """)


def restart_disclaimer():
    """Disclaimer for user if code hits restart at particular points"""
    print("""
    this code can sometimes raise such errors.
    if some person is able to assign classes to the department manually,
    then it is recommended that the maximum classes of
    the teachers in that department be increased
    """.upper())


def out_of_slots():
    """Print error to user if there are discrepancies in code for classes"""
    print("Error! Class has run out of free slots.")
    print("Too many subjects are being assigned.")
    print("Contact necessary faculty to edit the code and resolve the issue.")
    input("Press enter to exit: ")
    exit()


def time_table_instruction():
    """Print instruction for user on layout of time table"""
    print("INSTRUCTIONS")
    print("All time tables have 7 rows and 8 columns.")
    print("The first row is for Monday, second row for Tuesday and so on till
Saturday.")
    print("The first column is for the first period, second for second period
and so on till eighth period.")
    print()
```

Contents of *objects.py*

```
"""Contains the objects used in the program"""
from __future__ import annotations
from package.display import display_table


# TABLE: Base layout for all tables
# Contains 1 to 47 positions from monday to saturday
# First slot is empty so that positions and indexing are same
TABLE = [""] + [str(i) for i in range(1, 48)]


class Teacher:
    """
    Represents a teacher.
    Attributes:
        i_d: int
            teacher ID, must be unique
        f_name: str
```

```
19              teacher's first name
20          l_name: str
21              teacher's last name
22          subject: str
23              subject taught by the teacher
24          grades: tuple(int)
25              classes taught by the teacher (eg. 6th, 7th)
26          min_c: int
27              minimum classes to be taken by the teacher
28          max_c: int
29              maximum classes to be taken by the teacher
30          assigned_classes: dict
31              classes that have been assigned to the teacher
32          schedule: list
33              teacher's time table
34
35      """
36      def __init__(self, i_d: int, f_name: str, l_name: str, subject: str,
    grades: tuple[int], min_c: int, max_c: int):
37          self.i_d = i_d
38          self.f_name = f_name
39          self.l_name = l_name
40          self.subject = subject
41          self.grades = grades
42          self.assigned_classes = {}
43          self.schedule = list(TABLE)
44          self.min_c = min_c
45          self.max_c = max_c
46
47      def new_assigned_class(self, clss: Grade):
48          """Update details when teacher is assigned a class
49          Parameters
50              clss: the class that has been assigned (Grade object)
51          """
52          self.assigned_classes[str(clss.grade) + clss.section] = clss
53
54      def update(self, pos: str | int, grade: int, section: str):
55          """Update a position in schedule"""
56          self.schedule[int(pos)] = str(grade) + section
57
58      def print_schedule(self):
59          """Print the teacher schedule/time table"""
60          s = []
61          for pos in self.schedule:
62              # Show numbers as empty spaces
63              if pos.isdigit():
64                  s.append("")
65              else:
66                  s.append(pos)
67
68          print(self.f_name, self.l_name)
69          if self.schedule != TABLE:  # Print schedule only if not empty
70              display_table(s)
71          else:
72              print("Schedule is empty.")
73
```

```
    def is_free(self, *args: str) -> bool:
        """Check if teacher is free for all positions in args"""
        return all(self.schedule[int(pos)].isdigit() for pos in args)

    def reset_schedule(self):
        self.schedule = list(TABLE)

    def reset_all(self):
        self.schedule = list(TABLE)
        self.assigned_classes = {}


# Class object handles all class details
class Grade:
    """
    Represents a class.
    Attributes:
        grade: int
            grade of the class
        section: str
            section of the class
        main_subs: tuple(str)
            the main subjects of the class
        add_subs: dict(str: int)
            keys: additional subjects of the class, values: frequency per week
        grades: tuple(int)
            classes taught by the teacher (eg. 6th, 7th)
        faculty: dict(str: Teacher)
            all the class's faculty stored in dictionary
        schedule: list
            class's time table
    """
    def __init__(self, grade: int, section: str, main_subs: tuple[str],
add_subs: dict[str, int]):
        self.grade = grade
        self.main_subs = list(main_subs)
        self.add_subs = add_subs
        self.section = section
        self.faculty: dict[str, Teacher | tuple[Teacher]] = {}
        self.schedule = list(TABLE)
        if self.grade <= 5:             # if it is lower classes
            del self.schedule[41:]      # do not work on saturday
            self.schedule[17] = "CCA"  # CCA on wed. for primary classes
        else:
            self.schedule[33] = "CCA"  # CCA on Fri. for higher classes
            if self.grade in {11, 12}:
                self.schedule[46] = "WE"
                self.schedule[47] = "WE"

    def display_schedule(self):
        """Print the schedule of the class"""
        print(f"{self.grade}{self.section} Time Table")
        display_table(self.schedule)

    def print_faculty(self):
        """Print the faculty of the class"""
```

```
129         print(self.grade, self.section)
130         for s in self.faculty:
131             if isinstance(self.faculty[s], tuple):
132                 for t in self.faculty[s]:
133                     print(f"{s}: {t.f_name} {t.l_name}")
134             else:
135                 t = self.faculty[s]
136                 print(f"{s}: {t.f_name} {t.l_name}")
137 
138     def is_free(self, *args: str) -> bool:
139         """Check if class is free for all positions in args"""
140         return all(self.schedule[int(pos)].isdigit() for pos in args)
141 
142     def update(self, pos: str, sub: str):
143         self.schedule[int(pos)] = sub
144 
145     def display_all_details(self):
146         self.print_faculty()
147         self.display_schedule()
148 
149     def reset_schedule(self):
150         """Resets the schedule of the class.
151         If class has been assigned to any teacher in the faculty,
152         then remove the class from that teacher's schedule."""
153         clss = str(self.grade) + self.section
154         for t in self.faculty.values():
155             if isinstance(t, tuple):  # computer sub has multiple teachers
156                 for k in t:
157                     for ind, pos in enumerate(k.schedule):
158                         if pos == clss:
159                             k.schedule[ind] = str(ind)
160             else:
161                 for ind, pos in enumerate(t.schedule):
162                     if pos == clss:
163                         t.schedule[ind] = str(ind)
164 
165         self.schedule = list(TABLE)
166         if self.grade <= 5:
167             del self.schedule[41:]
168             self.schedule[17] = "CCA"
169         else:
170             self.schedule[33] = "CCA"
171 
172     def reset_all(self):
173         self.schedule = list(TABLE)
174         self.faculty = {}
175         if self.grade <= 5:             # if it is lower classes
176             del self.schedule[41:]      # do not work on saturday
177             self.schedule[17] = "CCA"  # CCA on Wed. for primary classes
178         else:
179             self.schedule[33] = "CCA"  # CCA on Fri. for higher classes
180             if self.grade in {11, 12}:
181                 self.schedule[46] = "WE"
182                 self.schedule[47] = "WE"
```

Contents of *inputs.py*

```
"""Input functions used in various tasks; input validation"""
from __future__ import annotations
ALL_SUBJECTS = (  # all subjects taught at the school
    "Maths",
    "English",
    "Hindi",
    "Tamil",
    "Sanskrit",
    "Evs",
    "Science",
    "Sst",
    "Computers",
    "Craft",
    "Morals",
    "Library",
    "Pt",
    "Drawing",
    "Yoga",
    "Music",
    "Economics",
    "Commerce",
)


def input_to_delete(data: list[list[str]]) -> int:
    """Input the ID of the teacher to be deleted from file"""
    all_ids = [row[0] for row in data]
    print("Now you will be deleting a teacher.")
    num = input("Enter ID of the teacher or Q to exit: ").capitalize()
    while True:
        if num.isdigit() and num in all_ids:
            break
        elif num == "Q":
            return False
        print("Invalid Input!")
        num = input("Enter: ").capitalize()
    return int(num)


def input_to_update(data: list[list[str]]) -> int:
    """Input the ID of the teacher to be updated in file"""
    all_ids = [row[0] for row in data]
    print("Now you will be updating a teacher.")
    num = input("Enter ID of the teacher: ").capitalize()
    while True:
        if num.isdigit() and num in all_ids:
            break
        print("Invalid Input!")
        num = input("Enter: ").capitalize()
    return int(num)


def input_subject() -> str:
    """Input the subject taken by new teacher to be added"""
```

```
55.     print("""
56.     Next, you will be entering the subject of the teacher.
57.     Please enter any of the following options.
58.     """)
59.     for sub in ALL_SUBJECTS:
60.         print(f"\t{sub}")
61.     print()
62.     subject = input("Enter subject: ").capitalize()
63.     while subject not in ALL_SUBJECTS:
64.         print("Invalid Input!")
65.         subject = input("Enter subject: ").capitalize()
66. 
67.     return subject
68. 
69. 
70. def input_id(data: list[list[str]]) -> int:
71.     data = [row for row in data if row]
72.     i_d = input("Enter ID of your new teacher: ")
73.     while True:
74.         if i_d.isdigit():
75.             for row in data:
76.                 if int(row[0]) == int(i_d):
77.                     print(f"ID already exists and in use by {row[1]}
    {row[2]}")
78.                     i_d = input("Enter again: ")
79.                     break
80.             else:
81.                 break
82.         else:
83.             print("ID can only be a number.")
84.             i_d = input("Enter again: ")
85. 
86.     return int(i_d)
87. 
88. 
89. def input_grades() -> list[int]:
90.     """Input the grades/classes taken by the new teacher to be added"""
91.     grades = []
92.     while True:
93.         grade = input("Enter grade or \"Q\" to exit: ").capitalize()
94.         if grade.isdigit() and int(grade) not in range(1, 13):
95.             print("Please enter a grade between 1 and 12!!")
96.             continue
97.         if not grade.isdigit() and grade != "Q":
98.             print("Invalid Input")
99.             continue
100.         if grade.isdigit() and int(grade) in grades:
101.             print("Grade already entered")
102.             continue
103.         if grade.isdigit():
104.             grades.append(int(grade))
105.         if grade == "Q" and grades:
106.             break
107.     return grades
108. 
109. 
```

```
110.def input_min_max_classes() -> tuple[int, int]:
111.    """Input min and max classes to be taken by new teacher to be added"""
112.    while True:
113.        min_c = input("Enter minimum classes taken by the teacher: ")
114.        while min_c.isdigit() is False:
115.            print("Please enter only numbers!!")
116.            min_c = input("Enter minimum classes taken by the teacher: ")
117.
118.        max_c = input("Enter maximum classes the teacher can take: ")
119.        while max_c.isdigit() is False:
120.            print("Please enter only numbers!!")
121.            max_c = input("Enter maximum classes to be taken by the teacher:
    ")
122.
123.        if int(max_c) >= int(min_c):
124.            break
125.        print("Max value must be greater than min value!!!!")
126.    return min_c, max_c
127.
128.
129.def yes_or_no(msg: str = "") -> str:
130.    """Take a yes or no input from the user and validate input"""
131.    choice = input(f"{msg} Y/N: ").upper()
132.    while choice not in ("Y", "N"):
133.        print("Invalid Input!")
134.        choice = input("Please enter Y/N: ").upper()
135.    return choice
136.
137.
138.def input_file_name() -> str:
139.    """asks user to input only a valid file name; check for illegal chars"""
140.    name = input("Please enter the file name of your output file: ")
141.    illegal = ("\\", "*", "/", "?", "|", "<", ">", "\"", ":")
142.    while True:
143.        for char in name:
144.            if char in illegal:
145.                print(f"You entered an illegal character {char}")
146.                name = input("Please enter a valid file name: ")
147.                break
148.        else:
149.            break
150.
151.    return name
```

Contents of *science.py*

```
1.  """Contains all operations related to modifying data in science_streams.csv"""
2.  from __future__ import annotations
3.  from package.files.connection import open_file
4.  import csv
5.
6.  FILE_NAME = "science_streams.csv"
7.
8.
9.  def verify(teachers: list):
10.     """Verify whether all teachers in parameter are already present in file"""
```

```
    with open_file(FILE_NAME, "r") as f:
        data = list(csv.reader(f))
    row_count = sum(1 for row in data if row)  # count no. of rows in file
    tot_sci_t = len(teachers)                  # count no. of science teachers
    return row_count == tot_sci_t              # return True if both equal


def read(teachers: list) -> tuple[list, list, list]:
    """
    Read the data stored in file into main program
    Data in file contains information about the teacher's streams
    Function must be called only if verify() returns True
    """
    physics = []
    chemistry = []
    biology = []
    with open_file(FILE_NAME, "r") as f:
        data = list(csv.reader(f))
    data = [row for row in data if row]  # remove any empty rows
    for row in data:
        idt, stream = row                # idt is id of teacher from data in
file
        for t in teachers:
            if t.i_d == int(idt):        # if teacher id is present in file,
                if stream == "P":        # classify Teacher object to
respective       stream
                    physics.append(t)
                elif stream == "C":
                    chemistry.append(t)
                elif stream == "B":
                    biology.append(t)

    return physics, chemistry, biology


def write(teachers: list) -> tuple[list, list, list]:
    """
    Write the science teachers data into the file and update in main program
    teachers parameter must be a list of Teacher objects
    returns tuple of lists of Teacher objects
    function must be called only if verify() returns False
    """
    f = open_file(FILE_NAME, "w")
    writer = csv.writer(f)
    # initialize lists for the three streams
    physics = []
    chemistry = []
    biology = []
    for t in teachers:
        print("Key:\nC: Chemistry, B: Biology, P: Physics")
        print(f"ID {t.i_d}. {t.f_name} {t.l_name} is a Science Teacher")
        stream = input("Enter the stream. P, C or B: ").upper()
        while True:  # while loop for input validation
            if stream in ("P", "C", "B"):
                break
            print("Please enter a valid choice")
```

```
            stream = input("Enter the stream. P, C, B: ").upper()

        writer.writerow((t.i_d, stream))
        # append the teachers to their respective lists
        if stream == "P":
            physics.append(t)
        elif stream == "C":
            chemistry.append(t)
        elif stream == "B":
            biology.append(t)
    f.close()
    return physics, chemistry, biology
```

Contents of *commerce.py*

```
"""Contains all functions that perform all file operations
related to commerce_streams.csv"""
from package.files.connection import open_file
from package.inputs import yes_or_no
import csv


FILE_NAME = "commerce_streams.csv"


def verify(teachers: list) -> bool:
    """Verifies if all teachers present in parameter teachers
    is present in the file"""
    with open_file(FILE_NAME, "r") as f:
        data = list(csv.reader(f))
    row_count = sum(1 for row in data if row)
    tot_com_t = len(teachers)
    return row_count == tot_com_t


def read(teachers: list) -> tuple:
    """
    Read the data stored in file into main program
    Data in file contains information about the teacher's streams
    Function must be called only if verify() returns True
    """
    entre = []
    business = []
    accounts = []
    with open_file(FILE_NAME, "r") as f:
        data = list(csv.reader(f))
    data = [row for row in data if row]
    for row in data:
        idt, streams = row
        streams = streams.split()
        for t in teachers:
            if t.i_d == int(idt):
                for stream in streams:
                    if stream == "E":
                        entre.append(t)
```

```
                    elif stream == "B":
                        business.append(t)
                    elif stream == "A":
                        accounts.append(t)

    return accounts, business, entre


def write(teachers: list) -> tuple:
    """
    Write the science teachers data into the file and update in main program
    teachers parameter must be a list of Teacher objects
    returns tuple of lists of Teacher objects
    function must be called only if verify() returns False
    """
    f = open_file(FILE_NAME, "w")
    writer = csv.writer(f)
    entre = []
    business = []
    accounts = []
    for t in teachers:
        streams = []
        print(f"ID {t.i_d}. {t.f_name} {t.l_name} is a Commerce Teacher")
        for stream in ["Accounts", "Business Studies", "Entrepreneurship"]:
            ch = yes_or_no(f"Does {t.f_name} {t.l_name} teach {stream}?")
            if ch == "Y":
                streams.append(stream[0])  # first letter is sufficient
        writer.writerow((t.i_d, " ".join(streams)))
        for stream in streams:
            if stream == "E":
                entre.append(t)
            elif stream == "B":
                business.append(t)
            elif stream == "A":
                accounts.append(t)
    f.close()
    return accounts, business, entre
```

Contents of *convert_to_pdf.py*

```
import os
from package.files.connection import open_file
try:
    from fpdf import FPDF
except ModuleNotFoundError:
    # if user does not have the module installed
    # run the required command and install it
    print("You do not have the fpdf module installed.")
    print("This module is required to provide output in pdf format.")
    input("Hit enter to start installation")
    while True:
        try:
            os.system("pip install fpdf")  # run command on command line
            from fpdf import FPDF
            print("Module successfully installed.")
```

```
            input("Press enter to continue")
            break
        except ModuleNotFoundError:
            print("ERROR!!!")
            print("Some Error has occurred.")
            print("Are you connected to the internet?")
            print("Please check your internet connection.")
            input("Hit enter to try again")


def create_pdf(filename, mode="P", font="Courier", siz=8.5, f=__file__):
    """Convert a text file into pdf format"""
    loc = os.path.realpath(os.path.join(os.getcwd(), os.path.dirname(f)))
    OUTPUT_PATH = os.path.join(loc, f"{filename}.pdf")
    pdf = FPDF()
    pdf.add_page(mode)  # add a page

    # set style and size of font for pdf
    pdf.set_font(font, size=siz)

    f = open_file(f"{filename}.txt", "r", fil=f)
    text = f.readlines()

    count = 0  # counter to check how many schedules are written on one page
    for i, line in enumerate(text, start=1):
        if line.startswith("SUBSTITUTION FOR TEACHER"):  # given string
denotes a new schedule
            if count == 0:  # for the first schedule, take a fresh page
                pdf.add_page(mode)  # count will never become 0 again
            count += 1  # increase counter for every schedule written
        if count > 4:  # do not write more than 4 schedules on one page
            pdf.add_page(mode)  # add a new page and write the next one
            count = 1  # reset the counter

        # create a cell
        pdf.cell(2000, 2.5, txt = line, ln = i, align = 'L')

    pdf.output(OUTPUT_PATH)  # save the pdf
```

Contents of *teacher.py*

```
"""Functions that perform file operations related to teacher_info.csv"""
from package.files.connection import open_file
from package import inputs as ip
import csv


def read(data: "list[list[str]]"):
    """Print all the teacher data from list data neatly"""
    lst = ["ID", "Name", "Subject", "Grades", "Min classes", "Max classes"]
    print(f"\n{lst[0]:>4} {lst[1]:<25} {lst[2]:<12} {lst[3]:<20} {lst[4]:^12}
{lst[5]:^12}")
    for row in data:
        name = row[1] + " " + row[2]
```

```
13.         print(f"{row[0]:>4}. {name:<25} {row[3]:<12} {row[4]:<20} {row[5]:^12}
    {row[6]:^12}")
14.
15.
16. def remove():
17.     """
18.     Remove a teacher from the file teacher_info.csv
19.     If user wants to exit without removing a teacher, ValueError is raised
20.     """
21.     f = open_file("teacher_info.csv", "r+")
22.     data = csv.reader(f)
23.     data = list(data)
24.     data = [row for row in data if row]
25.     read(data)
26.     num = ip.input_to_delete(data)
27.     if num is False:
28.         raise ValueError
29.     for row in data:
30.         if int(row[0]) == num:
31.             break
32.
33.     print("The following data is going to be deleted.")
34.     read([row])
35.     ch = input("Do you wish to continue? y/n: ").lower()
36.     while ch not in ("y", "n"):
37.         ch = input("Invalid input! Enter again: ")
38.     if ch == "n":
39.         raise ValueError
40.
41.     data.remove(row)
42.     f.close()
43.     f = open_file("teacher_info.csv", "w")
44.     writer = csv.writer(f)
45.     for row in data:
46.         writer.writerow(row)
47.     f.close()
48.     input("Data deleted successfully. Hit enter to continue")
49.
50.
51. def add():
52.     """Add a teacher to the file teacher_info.csv"""
53.     f = open_file("teacher_info.csv", "r+")
54.     i = 1
55.     while True:
56.         print(f"Enter details of teacher {i}")
57.         f_name = input("Enter first name: ")
58.         l_name = input("Enter last name: ")
59.         data = list(csv.reader(f))
60.         i_d = ip.input_id(data)
61.         subject = ip.input_subject()
62.         grades = ip.input_grades()
63.         grades = " ".join(str(grade) for grade in grades)
64.         min_classes, max_classes = ip.input_min_max_classes()
65.         row = (i_d, f_name, l_name, subject, grades, min_classes, max_classes)
66.         f.write("\n")
67.         writer = csv.writer(f)
```

```
        writer.writerow(row)
        choice = ip.yes_or_no("Any more teacher details to enter?")
        if choice == "Y":
            i += 1
        else:
            f.close()
            break


def update():
    """Update the details of an existing teacher in teacher_info.csv"""
    f = open_file("teacher_info.csv", "r")
    data = list(csv.reader(f))
    data = [i for i in data if i]  # remove empty rows
    read(data)
    while True:
        num = ip.input_to_update(data)
        for row in data:
            if int(row[0]) == num:
                print(f"You are editing data for ID {row[0]}: {row[1]}
{row[2]}")
                grades = ip.input_grades()
                min, max = ip.input_min_max_classes()
                grades = " ".join(str(g) for g in grades)
                row[4], row[5], row[6] = grades, min, max
                break
        else:
            print("Invalid input. The ID you entered is not present in the
file. Try again.")
            continue
        break
    f.close()
    f = open_file("teacher_info.csv", "w")
    writer = csv.writer(f)
       writer.writerows(data)
       f.close()
```

Contents of *general.py*

```
"""Functions that do various tasks used inside of main script"""

TABLE = [""] + [str(i) for i in range(1, 48)]


def positions_valid(*args: str):
    """Check if given slot is free for both teacher and classroom"""
    return not any(
        not c.is_free(pos)
        or not t.schedule[int(pos)].isdigit()
        for c, t, pos in args
    )


def teachers_free(pos: str, *args):
    """Check if teacher(s) is(are) free for a given position"""
    return all(t.schedule[int(pos)].isdigit() for t in args)
```

```
18.
19.
20. def sort_dict(d: dict) -> dict:
21.     """Sort dictionary by values"""
22.     keys = list(d.keys())
23.     values = list(d.values())
24.     number = [len(t) for t in d.values()]  # count number of teachers per slot
25.     for i in range(len(number)):  # initialize bubble sort
26.         for j in range(len(number) - i - 1):
27.             if number[j] > number[j+1]:  # if more teachers, push slot to end
    of list
28.                 keys[j], keys[j+1] = keys[j+1], keys[j]          # swap keys
29.                 values[j], values[j+1] = values[j+1], values[j]  # swap values
30.
31.     d = dict(zip(keys, values))
32.     return d
33.
34.
35. def get_key(clss, t):
36.     """Access a key from dictionary using value"""
37.     key_list = list(clss.faculty.keys())
38.     val_list = list(clss.faculty.values())
39.     return key_list[val_list.index(t)]
40.
41.
42. def all_different(slot):
43.     """Check if all positions in slot are on different days"""
44.     slot = set(slot)
45.     for i in range(6):
46.         day = set(TABLE[8*i+1: 8*i+9])
47.         if len(slot.intersection(day)) > 1:
48.             return False
49.     return True
50.
51.
52. def remove_values(d: dict, n: int) -> dict:
53.     """Removes the first n values from the dictionary d"""
54.     return dict(tuple(d.items())[n:])
```

Contents of *class_details.py*

```
1. from package.objects import Grade
2.
3.
4. def class_11_12_details(grade: '11 | 12', section: str) -> Grade:
5.     """
6.     Function storing details of classes 11th and 12th.
7.     Returns Grade object with necessary details
8.     """
9.
10.     if section == "A":
11.         return Grade(
12.                     grade,
13.                     "A",
14.                     ("Chemistry", "Physics", "Maths", "English", "Computers"),
```

```
                    {"Pt": 2, "Practicals": 4, "Library": 1},
                )
    elif section == "B":
        return Grade(
                    grade,
                    "B",
                    ("Chemistry", "Physics", "Maths", "English", "Biology"),
                    {"Pt": 2, "Practicals": 6, "Library": 1},
                )
    elif section == "C":
        return Grade(
                    grade,
                    "C",
                    ("Accounts", "Business", "Maths", "English", "Economics"),
                    {"Pt": 2, "Library": 1},
                )
    elif section == "D":
        return Grade(
                    grade,
                    "D",
                    ("Accounts", "Business", "Entre", "English", "Economics"),
                    {"Pt": 2, "Library": 1},
                )


def class_9_10_details(grade: '9 | 10', section: str) -> Grade:
    """
    Function storing details of classes 9th and 10th
    returns Grade object with necessary details
    """
    return Grade(grade,
                 section,
                 ("Science", "Maths", "English", "Language", "Sst"),
                 {
                  "Pt": 2,
                  "Craft": 2,
                  "Computers": 2,
                  "Practicals": 2,
                  "Yoga": 1,
                  "Morals": 1,
                  "Drawing": 1,
                  "Music": 1,
                  "Library": 1,
                 })


def class_6_7_8_details(grade: '6 | 7 | 8', section: str) -> Grade:
    """
    Function storing details of classes 6th, 7th, and 8th
    returns Grade object with necessary details
    """
    return Grade(grade,
                 section,
                 ("Science", "Maths", "English", "Sst"),
                 {
                  "Hindi": 5,
```

```
                    "Tamil": 5,
                    "Pt": 2,
                    "Craft": 2,
                    "Computers": 2,
                    "Yoga": 1,
                    "Morals": 1,
                    "Drawing": 1,
                    "Music": 1,
                    "Library": 1,
                   })


def class_4_5_details(grade: '4 | 5', section: str) -> Grade:
    """
    Function storing details of classes 4th and 5th
    returns Grade object with necessary details
    """
    return Grade(grade,
                 section,
                 ("Maths", "English", "Evs"),
                 {
                  "Hindi": 3,
                  "Tamil": 3,
                  "Pt": 2,
                  "Craft": 2,
                  "Computers": 2,
                  "Yoga": 1,
                  "Morals": 1,
                  "Drawing": 1,
                  "Music": 1,
                    })
```

Contents of *time_table.py*

```
"""Main script time_table.py"""
from __future__ import annotations
import sys
import os
import random
import csv
from itertools import permutations

from package.inputs import input_file_name
from package.files.connection import open_file
from package.objects import Teacher, Grade, TABLE
import package.display as dp
import package.commerce as com
import package.science as sci
import package.teacher as tc
import package.general as gn
import package.convert_to_pdf as pdf
import package.class_details as cdt


# all global variables
all_teachers: list[Teacher] = []
all_classes: dict[str, Grade] = {}
```

```
physics_lab = list(TABLE)
chemistry_lab = list(TABLE)
biology_lab = list(TABLE)
comp_lab_sen = list(TABLE)
comp_lab_jun = list(TABLE)

# store teachers with extra specialisations
physics_teachers = []
chemistry_teachers = []
biology_teachers = []
entre_teachers = []
business_teachers = []
accounts_teachers = []


class LackOfTeachersError(Exception):
    """Error raised when code identifies shortage of teachers"""


class RestartRequiredError(Exception):
    """Error raised when code is required to restart"""


class GoToMainMenuError(Exception):
    """Error raised when user needs to return to main menu"""


def reset_globals():
    """Reset all global variables. Call when restarting main()."""
    global physics_lab, chemistry_lab, biology_lab, comp_lab_jun, comp_lab_sen
    physics_lab = list(TABLE)
    chemistry_lab = list(TABLE)
    biology_lab = list(TABLE)
    comp_lab_sen = list(TABLE)
    comp_lab_jun = list(TABLE)
    for teacher in all_teachers:
        teacher.reset_all()
    for clss in all_classes.values():
        clss.reset_all()


def delete_globals():
    """Delete all data stored in global variables but not variable itself"""
    global all_classes, all_teachers
    reset_globals()
    # del lst[:] deletes all data from list without deleting variable name
    del biology_teachers[:]
    del chemistry_teachers[:]
    del physics_teachers[:]
    del business_teachers[:]
    del entre_teachers[:]
    del accounts_teachers[:]
    all_classes = {}
    all_teachers = []


```

```
80. def change_teacher_data():
81.     """Update, Remove, or Add teachers to teacher.csv file"""
82.     print("A: Add a teacher\nR: Remove a teacher\nU: Update a teacher info")
83.     ch = input("Enter your choice: ").capitalize()
84.     while ch not in ("A", "R", "U"):  # Validate the input
85.         print("Invalid Input")
86.         ch = input("Enter: ").capitalize()
87.
88.     if ch == "R":
89.         try:
90.             tc.remove()
91.         except ValueError:  # ValueError if user doesn't want to remove
    teacher
92.             pass
93.     elif ch == "A":
94.         tc.add()
95.     elif ch == "U":
96.         tc.update()
97.
98.
99. def classify_science():
100.     """Classify science teachers into 3 streams"""
101.     # take only science teachers of higher classes
102.     sci_teachers = [t for t in all_teachers
103.                     if t.subject == "Science"
104.                     and set(t.grades).issubset({9, 10, 11, 12})
105.                     ]
106.     flag = sci.verify(sci_teachers)
107.     p, c, b = sci.read(sci_teachers) if flag else sci.write(sci_teachers)
108.     physics_teachers.extend(p)
109.     chemistry_teachers.extend(c)
110.     biology_teachers.extend(b)
111.
112.
113.def classify_commerce():
114.     """Classify commerce teachers into 3 streams"""
115.     com_teachers = [t for t in all_teachers if t.subject == "Commerce"]
116.     flag = com.verify(com_teachers)
117.     a, b, e = com.read(com_teachers) if flag else com.write(com_teachers)
118.     accounts_teachers.extend(a)
119.     business_teachers.extend(b)
120.     entre_teachers.extend(e)
121.
122.
123.def all_labs_free(pos) -> bool:
124.     """Check if all three science labs are free for two given positions"""
125.     return (
126.         physics_lab[int(pos)].isdigit()
127.         and chemistry_lab[int(pos)].isdigit()
128.         and biology_lab[int(pos)].isdigit()
129.     )
130.
131.
132.def is_teachers_filled(sub: str, clss_grade: int) -> bool:
133.     """Check if teachers are filled and cannot be assigned any classes"""
134.     count = sum(
```

```
        teacher.max_c / len(teacher.grades)
        for teacher in all_teachers
        if clss_grade in teacher.grades and teacher.subject == sub
    )
    return count < 4


def min_max(teachers: list[Teacher], clss: Grade) -> Teacher:
    """Select a teacher from sample teachers
    satisfies minimum classes first, then satisfies max. classes"""
    random.shuffle(teachers)

    # check minimum classes first
    for t in teachers:
        if t.min_c == len(t.assigned_classes):
            continue
        if (
            t.min_c > len(t.assigned_classes)
            # distribute teachers across their classes evenly
            and ("".join(t.assigned_classes)).count(str(clss.grade)) <
int(t.min_c/len(t.grades))
        ):
            return t

    # check maximum classes
    for t in teachers:
        if len(t.assigned_classes) < t.max_c:
            return t

    # if both min and max fail, check if teachers are filled
    if is_teachers_filled(t.subject, clss.grade):
        print(f"ERROR! Maximum Classes for {t.subject} department exceeded!")
        print("Please update existing details or enter new teacher.")
        dp.restart_disclaimer()
        input("Press enter to continue")
        raise LackOfTeachersError
    else:
        raise RestartRequiredError


def check_class_faculty(clss: Grade):
    """Check if a given class has been assigned all required teachers"""

    # check the main subjects
    for sub in clss.main_subs:
        if sub == "Language":
            continue  # language is assigned and checked separately
        try:
            clss.faculty[sub]
        except KeyError:
            print(f"ERROR!! Lack of Teachers For {sub} For Class
{clss.grade}")
            print("Please Add The Required Teacher.")
            input("Press enter to continue")
            raise LackOfTeachersError

```

```
    # check the additional subjects
    for sub in clss.add_subs:
        if sub == "Practicals":
            continue  # practicals is assigned and checked separately
        try:
            clss.faculty[sub]
        except KeyError:
            print(f"ERROR!! Lack of Teachers For {sub} For Class
{clss.grade}")
            print("Please Add The Required Teacher.")
            input("Press enter to continue")
            raise LackOfTeachersError


def sync_language(clss_1: Grade, clss_2: Grade):
    """Sync language classes for clss_1 and clss_2"""
    sub1, sub2, sub3 = "Sanskrit", "Hindi", "Tamil"

    if sub1 in clss_1.faculty:
        t1 = clss_1.faculty[sub1]
    else:
        # if class does not have the specified language
        # dummy teacher is created to avoid errors
        t1 = Teacher(None, None, None, None, None, None, None)
    if sub2 in clss_1.faculty:
        t2 = clss_1.faculty[sub2]
    else:
        t2 = Teacher(None, None, None, None, None, None, None)
    if sub3 in clss_1.faculty:
        t3 = clss_1.faculty[sub3]
    else:
        t3 = Teacher(None, None, None, None, None, None, None)

    for i in range(6):
        day = clss_1.schedule[8*i+1:8*i + 9]
        random.shuffle(day)
        day = [int(p) for p in day if p.isdigit()]  # ensures clss_1 is free
        if not day:
            raise RestartRequiredError
        for pos in day:
            if clss_2.is_free(pos) and gn.teachers_free(pos, t1, t2, t3):
                clss_1.update(pos, "Language")
                clss_2.update(pos, "Language")
                for t in t1, t2, t3:
                    t.update(pos, clss_1.grade, clss_1.section)
                break
        else:
            # all positions on all days are full, restart
            raise RestartRequiredError


def assign_one_teacher(clss: Grade, teachers: dict[str, list[Teacher]]) ->
None:
    """From an identified list of teachers, assign any one.
    Function must be called within assign_faculty or assign_faculty_together
    only after the correct set of teachers has been identified
```

```
    """
    for subject in teachers:
        if subject not in clss.faculty:
            if subject == "Computers":  # few classes have two computer
teachers
                t1 = random.choice(teachers[subject])
                if 12 in t1.grades:  # if teacher is most senior
                    clss.faculty[subject] = t1
                    t1.new_assigned_class(clss)
                    continue
                t2 = random.choice(teachers[subject])
                while t1 == t2 or 12 in t2.grades:
                    t2 = random.choice(teachers[subject])
                clss.faculty[subject] = (t1, t2)
                t1.new_assigned_class(clss)
                t2.new_assigned_class(clss)
            else:
                t = min_max(teachers[subject], clss)
                clss.faculty[subject] = t
                t.new_assigned_class(clss)


def assign_one_lab(clss: Grade) -> None:
    """Assign lab classes to 11th and 12th"""
    # take the required teachers
    physics = clss.faculty["Physics"]
    chemistry = clss.faculty["Chemistry"]
    if clss.section == "B":
        biology = clss.faculty["Biology"]
    days = [0, 1, 2, 3, 4]
    random.shuffle(days)
    # k values -> 0 & 1: assign phy and chem, 2: assign bio
    for k in range(3):
        classes_assigned = False  # check if classes for one k value are
assigned
        for i in days:
            day = clss.schedule[8*i+1: 8*i+9]
            for j in range(len(day) - 1):
                pos1 = day[j]
                pos2 = day[j + 1]
                if pos1.isdigit() and pos2.isdigit():
                    if k in {0, 1}:
                        # assign phy and chem labs at same time
                        if (physics_lab[int(pos1)].isdigit()
                            and physics_lab[int(pos2)].isdigit()
                            and physics.is_free(pos1, pos2)
                            and chemistry_lab[int(pos1)].isdigit()
                            and chemistry_lab[int(pos2)].isdigit()
                            and chemistry.is_free(pos1, pos2)):
                            for pos in pos1, pos2:
                                clss.update(pos, "P/C Lab")
                                physics.update(pos, clss.grade, clss.section)
                                physics_lab[int(pos)] = str(clss.grade) +
clss.section
                                chemistry.update(pos, clss.grade,
clss.section)
```

```
                                chemistry_lab[int(pos)] = str(clss.grade) +
clss.section
                            classes_assigned = True
                            break
                    elif k == 2 and clss.section == "A":
                        classes_assigned = True
                        break
                    elif k == 2 and clss.section == "B":
                        if (
                            biology_lab[int(pos1)].isdigit()
                            and biology_lab[int(pos2)].isdigit()
                            and biology.is_free(pos1, pos2)
                        ):
                            for pos in pos1, pos2:
                                clss.update(pos, "Bio Lab")
                                biology.update(int(pos), clss.grade,
clss.section)
                                biology_lab[int(pos)] = str(clss.grade) +
clss.section
                            classes_assigned = True
                            break
            if classes_assigned:
                days.remove(i)
                break
        else:
            classes_assigned = False
            break
    if not classes_assigned:
        raise RestartRequiredError


def assign_faculty(clss: Grade) -> None:
    """Assigns faculty for main subjects and additional subjects for a
class"""
    # Assign faculty for main subjects
    allowed_teachers = {}  # Dictionary of all possible main subject teachers
    for teacher in all_teachers:
        # first, for 11th and 12th, assign teachers of special subjects
        # science and commerce teachers are assigned
        if clss.grade in (11, 12):
            if clss.section in ["A", "B"]:
                allowed_teachers["Physics"] = [t for t in physics_teachers
                                               if clss.grade in t.grades]
                allowed_teachers["Chemistry"] = [t for t in chemistry_teachers
                                                 if clss.grade in t.grades]
                if clss.section == "B":
                    allowed_teachers["Biology"] = [t for t in biology_teachers
                                                   if clss.grade in t.grades]

            elif clss.section in ("C", "D"):
                # unlike science groups, here no need to check for t.grades
                # since commerce is available only for 11th and 12th
                allowed_teachers["Accounts"] = accounts_teachers
                allowed_teachers["Business"] = business_teachers
                if clss.section == "D":
                    allowed_teachers["Entre"] = entre_teachers
```

```
347.
348.         if teacher.subject in clss.main_subs and clss.grade in teacher.grades:
349.             allowed_teachers.setdefault(teacher.subject, [])
350.             allowed_teachers[teacher.subject].append(teacher)
351.
352.     assign_one_teacher(clss, allowed_teachers)
353.
354.     # Assign faculty for additional subjects
355.     allowed_add_teachers = {}  # Dict of possible additional subject teachers
356.     for teacher in all_teachers:
357.         if teacher.subject in clss.add_subs and clss.grade in teacher.grades:
358.             allowed_add_teachers.setdefault(teacher.subject, [])
359.             allowed_add_teachers[teacher.subject].append(teacher)
360.
361.     assign_one_teacher(clss, allowed_add_teachers)
362.     check_class_faculty(clss)
363.
364.
365. def assign_language_together(clss_1: Grade, clss_2: Grade) -> None:
366.     """Assign language classes commonly to clss_1, clss_2"""
367.     allowed_teachers = {}  # Dict of all possible lang teachers
368.     for teacher in all_teachers:
369.         if (teacher.subject in ("Hindi", "Tamil", "Sanskrit")
370.            and clss_1.grade in teacher.grades):
371.             allowed_teachers.setdefault("Language", [])
372.             allowed_teachers["Language"].append(teacher)
373.
374.     # identify and separate all language teachers
375.     for teachers in allowed_teachers.values():
376.         sanskrit, tamil, hindi = [], [], []
377.         for t in teachers:
378.             if t.subject == "Hindi":
379.                 hindi.append(t)
380.             elif t.subject == "Sanskrit":
381.                 sanskrit.append(t)
382.             elif t.subject == "Tamil":
383.                 tamil.append(t)
384.
385.         if clss_1.section not in "CD":  # sanskrit is available only for A and
    B sections
386.             if not sanskrit:
387.                 print(f"ERROR!! Lack of Teachers For Sanskrit For Class
    {clss_1.grade}")
388.                 print("Please Add The Required Teacher.")
389.                 input("Press enter to continue")
390.                 raise LackOfTeachersError
391.
392.             t1 = random.choice(sanskrit)
393.             clss_1.faculty["Sanskrit"] = t1
394.             clss_2.faculty["Sanskrit"] = t1
395.             t1.new_assigned_class(clss_1)
396.             t1.new_assigned_class(clss_2)
397.
398.         if not tamil:
399.             print(f"ERROR!! Lack of Teachers For Tamil For Class
    {clss_1.grade}")
```

```
            print("Please Add The Required Teacher.")
            input("Press enter to continue")
            raise LackOfTeachersError

        t2 = random.choice(tamil)
        clss_1.faculty["Tamil"] = t2
        clss_2.faculty["Tamil"] = t2
        t2.new_assigned_class(clss_1)
        t2.new_assigned_class(clss_2)

        if not hindi:
            print(f"ERROR!! Lack of Teachers For Hindi For Class
{clss_1.grade}")
            print("Please Add The Required Teacher.")
            input("Press enter to continue")
            raise LackOfTeachersError

        t3 = random.choice(hindi)
        clss_1.faculty["Hindi"] = t3
        clss_2.faculty["Hindi"] = t3
        t3.new_assigned_class(clss_1)
        t3.new_assigned_class(clss_2)


def assign_faculty_together(clss_1: Grade, clss_2: Grade) -> None:
    """Assign certain faculty common for two classes"""
    allowed_teachers = {}  # Dict of all possible main subject teachers
    # assign language first
    assign_language_together(clss_1, clss_2)
    for clss in clss_1, clss_2:
        allowed_teachers = {}
        for t in all_teachers:
            if (
                # language teachers have been assigned separately
                t.subject not in ("Hindi", "Tamil", "Sanskrit")
                and t.subject in clss.main_subs
                and clss.grade in t.grades
            ):
                allowed_teachers.setdefault(t.subject, [])
                allowed_teachers[t.subject].append(t)
        assign_one_teacher(clss, allowed_teachers)
    allowed_add_teachers = {}  # Dictionary of all possible add. sub. teachers
    for t in all_teachers:
        if (
            t.subject in clss_1.add_subs
            and clss_1.grade in t.grades
        ):
            allowed_add_teachers.setdefault(t.subject, [])
            allowed_add_teachers[t.subject].append(t)

    assign_one_teacher(clss_1, allowed_add_teachers)
    assign_one_teacher(clss_2, allowed_add_teachers)
    check_class_faculty(clss_1)
    check_class_faculty(clss_2)
```

```
455.def schedule_full_week(clss: Grade, subs: list[str]) -> None:
456.    """Schedule one class of each subject in list subs on each day for entire
    week"""
457.    DAYS = 5 if clss.grade in (1, 2, 3, 4, 5) else 6
458.    for i in range(DAYS):
459.        day = clss.schedule[8*i + 1: 8*i + 9]
460.        day = [p for p in day if p.isdigit()]  # take only free positions
461.        possible_slots = list(permutations(day, len(subs)))  # take all
    possibilities
462.        random.shuffle(possible_slots)
463.        for slots in possible_slots:
464.            for j, slot in enumerate(slots):
465.                # check if the slot is free for teacher as well
466.                teacher = clss.faculty[subs[j]]
467.                if not gn.positions_valid((clss, teacher, slot)):
468.                    flag = False
469.                    break
470.            else:
471.                flag = True
472.            if flag:
473.                break
474.        else:
475.            raise RestartRequiredError
476.
477.        # once free slot is identified, assign the classes
478.        for j, pos in enumerate(slots):
479.            sub = subs[j]
480.            teacher = clss.faculty[sub]
481.            clss.update(pos, sub)
482.            teacher.update(int(pos), clss.grade, clss.section)
483.
484.
485.def schedule_add_sub(clss: Grade, slots: list[str], sub: str) -> None:
486.    """Schedule an additional subject sub to the class clss"""
487.    t = clss.faculty[sub]
488.    freq = clss.add_subs[sub]
489.    slots: list[tuple[str]] = [i for i in permutations(slots, freq) if
    gn.all_different(i)]
490.    if not slots:
491.        dp.out_of_slots()
492.    random.shuffle(slots)
493.    for slot in slots:
494.        for pos in slot:
495.            if not (gn.teachers_free(pos, t) and clss.is_free(pos)):
496.                break
497.        else:
498.            for pos in slot:
499.                t.update(pos, clss.grade, clss.section)
500.                clss.update(pos, sub)
501.            break
502.    else:
503.        raise RestartRequiredError
504.
505.
506.def schedule_computer(clss: Grade, slots: list[str]) -> None:
507.    """Schedule computer labs for a particular class."""
```

```
    freq = clss.add_subs["Computers"]
    slots: list[tuple[str]] = [i for i in permutations(slots, freq) if
gn.all_different(i)]
    random.shuffle(slots)

    # check if class has two computer teachers
    if isinstance(clss.faculty["Computers"], tuple):  # if class has two
computer teachers
        t1, t2 = clss.faculty["Computers"]
        for slot in slots:
            for pos in slot:
                # check if teachers and lab are free
                if not (gn.teachers_free(pos, t1, t2) and
comp_lab_jun[int(pos)].isdigit()):
                    break
            else:
                for pos in slot:
                    t1.update(pos, clss.grade, clss.section)
                    t2.update(pos, clss.grade, clss.section)
                    clss.update(pos, "Computers")
                    comp_lab_jun[int(pos)] = str(clss.grade) + clss.section
                break
        else:
            # if not able to assign classes, restart
            raise RestartRequiredError
    else:  # executes if class has only one computer teacher
        # a single computer teacher would be assigned the senior lab
        t = clss.faculty["Computers"]
        for slot in slots:
            for pos in slot:
                if not (gn.teachers_free(pos, t) and
comp_lab_sen[int(pos)].isdigit()):
                    break
            else:
                for pos in slot:
                    t.update(pos, clss.grade, clss.section)
                    clss.update(pos, "Computers")
                    comp_lab_sen[int(pos)] = str(clss.grade) + clss.section
                break
        else:
            raise RestartRequiredError


def sync_common_subs(clss_1: Grade, clss_2: Grade, common_subs: list[str]) ->
None:
    """Sync subs in common_subs for both clss_1 and clss_2"""
    while True:
        for sub in common_subs:
            if sub in clss_1.main_subs:
                if sub == "Language":
                    sync_language(clss_1, clss_2)
            elif sub in clss_1.add_subs:
                if sub == "Practicals":
                    # if sub is practicals, assign labs and teachers
                    physics = random.choice(physics_teachers)
                    chemistry = random.choice(chemistry_teachers)
```

```
                    biology = random.choice(biology_teachers)
                    classes_assigned = False
                    days_num = [0, 1, 2, 3, 4]  # only weekdays should have practicals
                    random.shuffle(days_num)
                    for i in days_num:
                        # check each weekday if class can be assigned
                        day = clss_1.schedule[8*i+1:8*i + 9]
                        for j in range(len(day) - 1):
                            pos1 = day[j]
                            pos2 = day[j+1]
                            if pos1.isdigit() and pos2.isdigit():
                                # check if second class, teachers, and labs are free
                                if (
                                    clss_2.is_free(pos1, pos2)
                                    and all_labs_free(pos1)
                                    and all_labs_free(pos2)
                                    and gn.teachers_free(pos1, physics, chemistry, biology)
                                    and gn.teachers_free(pos2, physics, chemistry, biology)
                                ):
                                    for pos in pos1, pos2:
                                        for clss in clss_1, clss_2:
                                            clss.update(pos, sub)
                                            for lab in (physics_lab, biology_lab, chemistry_lab):
                                                if (
                                                    pos == pos1 and clss == clss_1
                                                    or pos == pos2 and clss == clss_2
                                                ):
                                                    lab[int(pos)] = str(clss.grade) + clss.section
                                                    for t in (physics, chemistry, biology):
                                                        t.update(int(pos), clss.grade, clss.section)

                                    classes_assigned = True
                                    break
                            break

                    if not classes_assigned:
                        # if classes not assigned, reset the schedules and try again
                        for clss in clss_1, clss_2:
                            remove_class_from_labs(clss.grade, clss.section)
                            clss.reset_schedule()
                        break
            else:
                break


```

```
604.def sync_bio_entre(clss_1: Grade, clss_2: Grade) -> None:
605.    """syncs biology and entrepreneurship classes
606.    clss_1 has to be the biology class and clss_2 has to be the
    entrepreneurship class
607.    """
608.    while True:
609.        generate(clss_1)  # generate entire time table for biology class
610.        t2 = clss_2.faculty["Entre"]
611.        for i, pos in enumerate(clss_1.schedule):
612.            if pos in ["Bio Lab", "Biology"]:
613.                # for positions where clss_1 has biology,
614.                # try assigning entre to clss_2 at same position
615.                if not gn.teachers_free(i, t2):
616.                    remove_class_from_labs(clss_1.grade, clss_1.section)
617.                    clss_1.reset_schedule()
618.                    clss_2.reset_schedule()
619.                    break
620.                else:
621.                    clss_2.update(i, "Entre")
622.                    t2.update(i, clss_2.grade, clss_2.section)
623.        else:
624.            break
625.        continue
626.
627.
628.def generate(clss: Grade) -> None:
629.    """Generate time table for one class independently"""
630.    if "Practicals" in clss.add_subs and clss.grade in (11, 12):
631.        assign_one_lab(clss)
632.
633.    if clss.grade not in (11, 12) or clss.section not in "D":
634.        schedule_full_week(clss, clss.main_subs)
635.    else:
636.        # 11D and 12D are scheduled using sync_bio_entre
637.        # so Entre subject must not be scheduled again
638.        schedule_full_week(clss, [s for s in clss.main_subs if s != "Entre"])
639.    for sub in clss.add_subs:
640.        slots = [p for p in clss.schedule if p.isdigit()]
641.        if sub == "Computers":
642.            schedule_computer(clss, slots)
643.        elif sub != "Practicals":
644.            if clss.add_subs[sub] >= 2:
645.                # if there are many multiple subs to assign
646.                # then do not use permutations
647.                # as it will take a long time
648.                t = clss.faculty[sub]
649.                count = 0
650.                while True:
651.                    pos = random.choice(slots)
652.                    if gn.teachers_free(pos, t):
653.                        clss.update(pos, sub)
654.                        t.update(int(pos), clss.grade, clss.section)
655.                        count = 0
656.                        slots.remove(pos)
657.                    freq = clss.add_subs[sub]
658.                    if clss.schedule.count(sub) == freq:
```

```
                        break
                    elif not slots:  # check if slots is empty
                        dp.out_of_slots()  # call the error msg

                    count += 1
                    if count == 1000:
                        raise RestartRequiredError
            else:
                # assign classes using permutation method
                schedule_add_sub(clss, slots, sub)

    # fill up remaining empty slots
    leave_free = 2 if clss.grade in (4, 5) else 0
    free_slots(clss, leave_free)


def generate_together(clss_1: Grade, clss_2: Grade, common_subs: list[str]) ->
None:
    """sync certain subjects together for two classes"""
    # schedule main and additional subs for both classes
    for clss in clss_1, clss_2:
        main_subs_to_assign = [s for s in clss.main_subs if s not in
common_subs]
        schedule_full_week(clss, main_subs_to_assign)
        for sub in clss.add_subs:
            all_available_slots = [p for p in clss.schedule if p.isdigit()]
            if sub in common_subs:
                pass
            elif sub == "Computers":
                schedule_computer(clss, all_available_slots)
            else:
                schedule_add_sub(clss, all_available_slots, sub)
        free_slots(clss)


def free_slots(clss: Grade, leave_free: int = 0) -> None:
    """Fill up leftover slots on time table"""
    sub_freq = {sub: 0 for sub in clss.main_subs}
    positions_free = {}
    for pos in clss.schedule:
        if pos.isdigit():
            available_teachers = []
            for s in list(clss.faculty):
                t = clss.faculty[s]
                if (
                    not isinstance(t, tuple)  # tuple in case of computer sub.
                    and s in clss.main_subs
                    and gn.teachers_free(pos, t)
                    and s != "Entre"  # Assigned separately in sync_bio_entre
                ):
                    available_teachers.append(t)
            if not available_teachers:
                # no teachers available for position pos
                raise RestartRequiredError
            random.shuffle(available_teachers)
            positions_free[pos] = available_teachers
```

```
713.
714.     positions_free = gn.sort_dict(positions_free)
715.     positions_free = gn.remove_values(positions_free, leave_free)
716.
717.     for pos in positions_free:
718.         if len(positions_free[pos]) == 1:
719.             t = positions_free[pos][0]
720.             sub = gn.get_key(clss, t)
721.             t.update(int(pos), clss.grade, clss.section)
722.             clss.update(pos, sub)
723.             sub_freq[sub] += 1
724.         else:
725.             for t in positions_free[pos]:
726.                 sub = gn.get_key(clss, t)
727.                 if sub_freq[sub] == min(sub_freq.values()):
728.                     t.update(int(pos), clss.grade, clss.section)
729.                     clss.update(pos, sub)
730.                     sub_freq[sub] += 1
731.                     break
732.             else:
733.                 t = random.choice(positions_free[pos])
734.                 t.update(int(pos), clss.grade, clss.section)
735.                 clss.update(pos, sub)
736.                 sub_freq[sub] += 1
737.
738.
739. def substitutions() -> None:
740.     """Assign substitution teachers in case a teacher is absent"""
741.     dp.substitution_instruction()  # display instruction for user
742.     dp.display_teachers(all_teachers)  # show all teachers with ID
743.     for t in all_teachers:
744.         assign_substitution_teachers(t)
745.
746.
747. def assign_substitution_teachers(teacher_to_be_subbed: Teacher) -> None:
748.     """
749.     Main function for ordering substitution teachers
750.     Teachers are first separated into primary and secondary classes
751.     (we cannot assign a primary teacher as substitute for secondary class)
752.     Teachers are ordered by a priority criteria
753.     Those who already teach the class are considered first
754.
755.     By default only four teachers are suggested
756.     but this number can be increased in MAX_TO_SHOW variable
757.     SIZE_OF_TABLE, which controls the output table size
758.     may need to be adjusted according to number of teachers displayed
759.     table size for SIZE_OF_TABLE > 2 will not fit in pdf output
760.     """
761.     MAX_TO_SHOW = 4
762.     SIZE_OF_TABLE = 2
763.     secondary = [t for t in all_teachers if min(t.grades) > 2]
764.     primary = [t for t in all_teachers if t not in secondary]
765.     print(f"\nSUBSTITUTION FOR TEACHER {teacher_to_be_subbed.f_name}
    {teacher_to_be_subbed.l_name}")
766.     substitution_table = list(TABLE)
767.     for ind, pos in enumerate(teacher_to_be_subbed.schedule):
```

```
768.         if pos.isdigit() is False and pos:
769.             try:
770.                 clss = teacher_to_be_subbed.assigned_classes[pos]
771.                 # create lists to order teachers by priority
772.                 priority_list = []
773.                 first_priority = []
774.                 second_priority = []
775.                 class_faculty = list(clss.faculty.values())
776.                 random.shuffle(class_faculty)
777.                 for t in class_faculty:
778.                     if isinstance(t, tuple):
779.                         for k in t:
780.                             if k.is_free(ind):
781.                                 second_priority.append(k)
782.                     elif t.subject in clss.main_subs:
783.                         # teachers who teach main subjects get highest
    priority
784.                         if t.is_free(ind):
785.                             first_priority.append(t)
786.                     elif t.is_free(ind):
787.                         # all other teachers receive second priority
788.                         second_priority.append(t)
789.
790.                 priority_list.extend(first_priority)
791.                 priority_list.extend(second_priority)
792.                 del first_priority, second_priority
793.                 if clss.grade < 3:
794.                     for t in primary:
795.                         if (
796.                             t not in priority_list
797.                             and t.is_free(ind)
798.                         ):
799.                             priority_list.append(t)
800.                 else:
801.                     for t in secondary:
802.                         if (
803.                             t not in priority_list
804.                             and t.is_free(ind)
805.                         ):
806.                             priority_list.append(t)
807.             except KeyError:
808.                 pass  # Practicals subject gives KeyError
809.
810.             else:
811.                 details = f"{clss.grade}{clss.section}: " + ", ".join(
812.                     str(t.i_d) for t in priority_list[:MAX_TO_SHOW]
813.                 )
814.
815.                 substitution_table[ind] = details
816.
817.     dp.display_table(substitution_table, SIZE_OF_TABLE)
818.
819.
820. def load_teacher_data() -> None:
821.     """Load all data from teacher_info.csv into main program"""
822.     f = open_file("teacher_info.csv", "r")
```

```
    data = list(csv.reader(f))
    f.close()
    for row in data:
        if row:
            i_d, f_n, l_n, sub, grades, min_c, max_c = row
            i_d, min_c, max_c = int(i_d), int(min_c), int(max_c)  # convert
str to int
            grades = tuple(int(g) for g in grades.split())  # convert grades
from str to tuple
            all_teachers.append(Teacher(i_d, f_n, l_n, sub, grades, min_c,
max_c))


def remove_class_from_labs(clss_grade: int, clss_section: str) -> None:
    """Remove a particular class from all the labs"""
    clss_value = str(clss_grade) + clss_section
    for lab in physics_lab, chemistry_lab, biology_lab:
        for index, pos in enumerate(lab):
            if pos == clss_value:
                lab[index] = str(index)


def initialize() -> None:
    """Load all data from files and create all objects"""
    load_teacher_data()
    classify_science()
    classify_commerce()
    # Making the classes with default subjects and frequency of extra subjects
    for clss in 6, 7, 8:
        for sec in "ABCD":
            all_classes[f"{clss}{sec}"] = cdt.class_6_7_8_details(clss, sec)
    for clss in 9, 10:
        for sec in "ABCD":
            all_classes[f"{clss}{sec}"] = cdt.class_9_10_details(clss, sec)
    for clss in 11, 12:
        for sec in "ABCD":
            all_classes[f"{clss}{sec}"] = cdt.class_11_12_details(clss, sec)


def main(choice: 0 | 1 | 2) -> bool:
    """The main function to be executed"""
    os.system("cls")
    if choice == 0:
        os.system("cls")
        choice = dp.home_screen()

    if choice == 1:
        change_teacher_data()
        delete_globals()
        initialize()
        raise GoToMainMenuError

    elif choice == 2:
        # assigning faculty for classes 11, 12
        for clss, obj in all_classes.items():
            if int(clss[:-1]) in (11, 12):
```

```
                assign_faculty(obj)

        # assigning faculty for classes 9, 10
        assign_faculty_together(all_classes["9A"], all_classes["9B"])
        assign_faculty_together(all_classes["9C"], all_classes["9D"])
        assign_faculty_together(all_classes["10A"], all_classes["10B"])
        assign_faculty_together(all_classes["10C"], all_classes["10D"])

        # assigning faculty for classes 6, 7, 8
        for clss, obj in all_classes.items():
            if int(clss[:-1]) in (6, 7, 8):
                assign_faculty(obj)

        # Generating Timetable for classes 6, 7, 8
        for i in (6, 7, 8):
            for sec in ("A", "B", "C", "D"):
                clss = all_classes[f"{i}{sec}"]
                generate(clss)

        # generating timetable for classes 9, 10
        sync_common_subs(all_classes["9A"], all_classes["9B"], ["Language",
    "Practicals"])
        sync_common_subs(all_classes["9C"], all_classes["9D"], ["Language",
    "Practicals"])
        sync_common_subs(all_classes["10A"], all_classes["10B"], ["Language",
    "Practicals"])
        sync_common_subs(all_classes["10C"], all_classes["10D"], ["Language",
    "Practicals"])
        generate_together(all_classes["9A"], all_classes["9B"], ["Language",
    "Practicals"])
        generate_together(all_classes["9C"], all_classes["9D"], ["Language",
    "Practicals"])
        generate_together(all_classes["10A"], all_classes["10B"], ["Language",
    "Practicals"])
        generate_together(all_classes["10C"], all_classes["10D"], ["Language",
    "Practicals"])

        # generating timetables for 11, 12 BD sections
        sync_bio_entre(all_classes["11B"], all_classes["11D"])
        sync_bio_entre(all_classes["12B"], all_classes["12D"])

        # generating timetables for 11, 12 AC sections
        for i in (11, 12):
            for sec in ("A", "C"):
                clss = all_classes[f"{i}{sec}"]
                generate(clss)

        # generating timetables for 11, 12 D section
        # this is done separately due to working of sync_bio_entre function
        for clss in "11D", "12D":
            generate(all_classes[clss])

        # input file name to write final output
        print("CAUTION!!!\nEntering a file name that already exists will
    result in overwriting of data in file")
        filename = input_file_name()
```

```
923.
924.         stdout_origin = sys.stdout  # save terminal stdout in stdout_origin
925.         sys.stdout = open_file(f"{filename}.txt", "w", fil=__file__)
926.         # Displaying the time table for classes
927.         dp.time_table_instruction()
928.         for clss in all_classes.values():
929.             clss.print_faculty()
930.             clss.display_schedule()
931.             print()
932.
933.         # Displaying timetable for labs
934.         print("Physics Lab")
935.         dp.display_table(physics_lab)
936.         print("Chemistry Lab")
937.         dp.display_table(chemistry_lab)
938.         print("Biology Lab")
939.         dp.display_table(biology_lab)
940.         print("Computer Lab Junior")
941.         dp.display_table(comp_lab_jun)
942.         print("Computer Lab Senior")
943.         dp.display_table(comp_lab_sen)
944.         for teacher in all_teachers:
945.             print()
946.             teacher.print_schedule()
947.         sys.stdout.close()
948.
949.         sys.stdout = open_file(f"{filename}_substitutions.txt", "w",
    fil=__file__)
950.         substitutions()
951.         sys.stdout.close()
952.         sys.stdout = stdout_origin
953.
954.         # create pdf format using text files created
955.         pdf.create_pdf(filename, f=__file__)
956.         pdf.create_pdf(f"{filename}_substitutions", "L", siz=7, f=__file__)
957.
958.         print('Time table successfully generated and saved.')
959.         print(f"Files saved in txt and pdf formats to location
    {os.path.dirname(__file__)}")
960.         input("Press enter to exit")
961.         return True
962.     elif choice == 3:
963.         return True
964.
965.
966. if __name__ == "__main__":
967.     mode = 0
968.     initialize()
969.     while True:
970.         try:
971.             flag = main(mode)
972.             if flag:
973.                 break
974.         except GoToMainMenuError:
975.             mode = 0
976.         except LackOfTeachersError:
```

```
977.             mode = 1
978.         except RestartRequiredError:
979.             mode = 2
980. 
981.         print("Loading...")
982.         reset_globals()
```

8. Output

1. Editing Teacher Data

   When the user edits the details of the file, there are various features we designed to ensure only the valid data is entered and various other fail safes are built in.

   a. Deleting a teacher from the file

      i. Main Menu

```
Welcome to the Time Table Generator!

        1. Edit teacher details
        2. Generate time table
        3. Exit

Please enter your choice: 1
```

      ii. User is asked whether they wish to add, remove, or update a teacher data

```
A: Add a teacher
R: Remove a teacher
U: Update a teacher info
Enter your choice:
```

      iii. Error Checking ensures the user enters only one of the desired inputs.

```
Enter your choice:
Invalid Input
Enter: Remove
Invalid Input
Enter: R
```

iv. The details of every teacher in the file are displayed, so that the user can check the details of the teacher they are about to delete.

```
ID Name                    Subject      Grades              Min classes  Max classes
 1. Somini Sudeep           English      10 11 12                  4            5
 2. Anupama                 English      11 12                     4            5
 3. Lavanya Lousia Tonia    English      9 10                      4            5
 4. Padmini M.S             English      8 9                       4            5
 5. Pritha                  English      6 7 8                     4            5
 6. Deepa M                 English      6 7                       4            5
 7. Gunjan Agarwal          English      4 5                       4            5
 8. Elangovan Jain          Hindi        8 9 10                    4            7
 9. Ashok Kumar Pandey      Hindi        5 6 7 8                   4            7
10. Jyoti Sahu              Hindi        5 6 7 8                   4            7
11. Neelima Devi            Hindi        4 5 6                     5            7
12. Sayee Lakshmi           Tamil        9 10                      4            6
13. Elavarasi P             Tamil        6 7 8                     4            5
14. Prema                   Tamil        6 7 8                     4            5
15. Vijayalakshmi Ramesh PK Tamil        6 7 8                     4            5
16. Ranganathan R           Sanskrit     9 10                      4            6
17. Geetha Ganesh           Maths        12                        2            2
18. Hemalatha P             Maths        11 12                     4            5
```

v. The user enters the ID of the teacher to delete, and also gets the option to cancel the operation if they wish to.

```
Now you will be deleting a teacher.
Enter ID of the teacher or Q to exit: 55
The following data is going to be deleted.

  ID Name                  Subject      Grades              Min classes  Max classes
  55. Vrinda Unnikrishnan   Craft        4 5 6                     6           10
Do you wish to continue? y/n: y
Data deleted successfully. Hit enter to continue
```

b. Adding a new teacher to the file

i. The user enters “A” to add a new teacher to the file.

```
A: Add a teacher
R: Remove a teacher
U: Update a teacher info
Enter your choice: A
```

ii. The user first enters the first and last names of the teacher.

```
Enter details of teacher 1
Enter first name: Sumathi
Enter last name: S
```

iii. The user then enters the unique ID for the teacher. They cannot enter an ID that is already in use by some other teacher.

```
Enter ID of your new teacher: 23
ID already exists and in use by Rajalekshmy Balaji
Enter again: 97
```

iv. The user enters the subject to be taught by the teacher. They can only input a subject present in the list printed to them. All other inputs are invalidated.

```
Next, you will be entering the subject of the teacher.
Please enter any of the following options.

    Maths
    English
    Hindi
    Tamil
    Sanskrit
    Evs
    Science
    Sst
    Computers
    Craft
    Morals
    Library
    Pt
    Drawing
    Yoga
    Music
    Economics
    Commerce

Enter subject: Environment
Invalid Input!
Enter subject: evs
```

v. The user enters the grades to be taught by the teacher. The number entered must be between 1 and 12 and they cannot enter the same number twice.

```
Enter grade or "Q" to exit: 2
Enter grade or "Q" to exit: 2
Grade already entered
Enter grade or "Q" to exit: 3
Enter grade or "Q" to exit: 4
Enter grade or "Q" to exit: 13
Please enter a grade between 1 and 12!!
Enter grade or "Q" to exit: q
```

vi. The user then enters the minimum and maximum classes that the teacher is allowed to take. Code ensures that minimum value is less than maximum.

```
Enter minimum classes taken by the teacher: 4
Enter maximum classes the teacher can take: 3
Max value must be greater than min value!!!!
Enter minimum classes taken by the teacher: 3
Enter maximum classes the teacher can take: 4
```

vii. If the user has more teacher details to enter, they are given the choice to do so, otherwise the program returns to the main menu.

```
Any more teacher details to enter? Y/N: n
```

c. Updating teacher details

i. The user enters “U” to update the existing details.

```
A: Add a teacher
R: Remove a teacher
U: Update a teacher info
Enter your choice: u
```

ii. Similar to deleting a teacher, the user is displayed all the existing data and asked to enter the ID to update.

```
Now you will be updating a teacher.
Enter ID of the teacher: 23
```

iii. Similar to adding a teacher, the user enters the grades to be taught by the teacher

```
Enter grade or "Q" to exit: 3
Enter grade or "Q" to exit: 4
Enter grade or "Q" to exit: 5
Enter grade or "Q" to exit: q
```

iv. Similar to adding a teacher, the user enters the minimum and maximum classes taught by the teacher.

```
Enter minimum classes taken by the teacher: 4
Enter maximum classes the teacher can take: 5
```

v. The user is notified when data is updated successfully. They are taken back to the main menu on pressing enter.

```
Data updated successfully. Hit enter to continue
```

2. Some other features
    a. Whenever the code identifies a shortage of teachers, the user is prompted to enter a teacher for the required subject.

```
Welcome to the Time Table Generator!

        1. Edit teacher details
        2. Generate time table
        3. Exit

Please enter your choice: 2
ERROR!! Lack of Teachers For Library For Class 11
Please Add The Required Teacher.
Press enter to continue
```

On hitting enter, the user is taken to the following screen and can then proceed accordingly.

```
A: Add a teacher
R: Remove a teacher
U: Update a teacher info
Enter your choice:
```

    b. Whenever a science or commerce teacher is removed or added to the file, the code requires the user to enter the specific subject taught by the teacher. This is illustrated as follows:
        i. Suppose we add a new science teacher to the file as follows

```
Enter details of teacher 1
Enter first name: New Science
Enter last name: Teacher

Enter subject: Science
Enter grade or "Q" to exit: 9
Enter grade or "Q" to exit: 10
Enter grade or "Q" to exit: 11
Enter grade or "Q" to exit: q
Enter minimum classes taken by the teacher: 4
Enter maximum classes the teacher can take: 5
Any more teacher details to enter? Y/N: n
```

ii. The code immediately takes the user through the following process

```
Key:
C: Chemistry, B: Biology, P: Physics
ID 24. Sreelatha Harish is a Science Teacher
Enter the stream. P, C or B: c
Key:
C: Chemistry, B: Biology, P: Physics
ID 25. Padma G is a Science Teacher
Enter the stream. P, C or B: c
Key:
C: Chemistry, B: Biology, P: Physics
ID 26. Vasanthy  is a Science Teacher
Enter the stream. P, C or B: b
Key:
C: Chemistry, B: Biology, P: Physics
ID 27. Flora Bhaskar is a Science Teacher
Enter the stream. P, C or B: b
Key:
C: Chemistry, B: Biology, P: Physics
ID 28. Vijaya Devanathan is a Science Teacher
Enter the stream. P, C or B: p
Key:
C: Chemistry, B: Biology, P: Physics
ID 29. Kalavathy T is a Science Teacher
Enter the stream. P, C or B: p
Key:
C: Chemistry, B: Biology, P: Physics
ID 100. New Science Teacher is a Science Teacher
Enter the stream. P, C or B: c
```

The user enters all the required details for each science teacher and then the user returns to the main menu. This step ensures data integrity by ensuring that whenever the science or commerce teachers in the file are changed, their specific streams are also taken into consideration. A similar process is followed for the commerce teachers. The data for these are stored in two files: *science_streams.csv* and *commerce_streams.csv*

3. If the user wants to exit the program, they can enter “3” in the main menu and the program ends.

```
Welcome to the Time Table Generator!

        1. Edit teacher details
        2. Generate time table
        3. Exit

Please enter your choice: 3
```

4. Generating the time tables

   i. On the main menu, the user must enter "2" to start generating the time tables. The entire process may take around 2 minutes.

```
Welcome to the Time Table Generator!

        1. Edit teacher details
        2. Generate time table
        3. Exit

Please enter your choice: 2
```

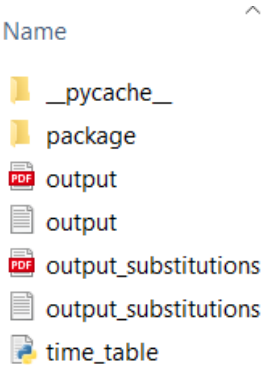
   ii. While the time tables are being generated, the user sees the message "Loading…"

```
Loading...
```

   iii. Final output: The user enters the file name and the command line shows the location to where the file is saved.

```
CAUTION!!!
Entering a file name that already exists will result in overwriting of data in file
Please enter the file name of your output file: output
Time table successfully generated and saved.
Files saved in txt and pdf formats to location d:\Time Table Generator
Press enter to exit
```

When the user navigates to the given location, the four output files can be found. Two of the files are in .txt format and two files are in .pdf format. One file is for the main time tables of all teachers and classes and the other file contains details regarding substitutions.



Important functions in our program

1. ***sync_language*()** - sync language subjects for two classes. Check if H, T, S teachers are free and both classes are free in a given position. Let the two classes be clss1 and clss2. First, we iterate through all the days of the week, and for each day of the week, we take all the positions where clss_1 is free. Then, we take one position at random. Then check if clss_2 and all three teachers are free, if yes then update the time tables for all.
2. ***assign_one_teacher()*** - we input a dictionary of {subject: list[Teacher]}. For computer classes, if we use the junior lab, assign two computer teachers. Else, assign only one.
3. ***assign_one_lab()*** - take the phy, chem, and bio teachers. Take any weekday at random and from that day, take two consecutive positions. Check if those two positions are free for both the class, the teacher, and the lab. First, we assign chem and phy, then we assign biology lab for the bio classes. Phy and chem labs are assigned at the same time to a particular class.
4. ***assign_faculty()*** - assign faculty for main subjects and additional subjects for one class. Make a dictionary, where key is the subject name, value is a list of teachers who can teach that sub for that particular class. Once the dictionary is prepared, call assign_one_teacher() func.
5. ***assign_language_together()*** - assigns language teachers commonly to two classes. Called within assign_faculty_together() function.
6. ***assign_faculty_together()*** - assign language commonly using the assign_language_together() func, then assign main subject and additional subject teachers in a similar manner as assign_faculty() function.
7. ***schedule_full_week()*** - part of generating all the time tables. We take a particular class, and assign subjects on every day of the week using permutation.
8. ***schedule_add_sub()*** - assign an additional subject with its required frequency using permutations method.
9. ***schedule_computer()*** - for a particular class, assign the computer classes and update the required lab correctly. If the class has two teachers, use the junior lab. Check if both teachers and the lab are free. If a class has one teacher, check if the senior lab is free and update the time tables accordingly.
10. ***sync_common_subs()*** - for two classes, schedule subjects commonly. Mainly, language and practical subjects are covered. For language, call the sync_language function. If the subject is practical, take one teacher from each science stream, take

any day of the week at random, and check if two consecutive positions are free for both the classes, all the three teachers and all the three labs.

11. ***sync_bio_entre()*** - sync the biology and entre classes. First, generate a time table for the bio class entirely. Once done, go to the second class, and schedule entrepreneurship class at the same time as bio classes are scheduled for the bio class.
12. ***generate()*** - Schedule an entire time table for one class. Various functions like, assign_one_lab, schedule_full_week, schedule_computer, schedule_add_subs, and free_slots all are called within this function.
13. ***generate_together*()** - function specifically for classes that require syncing of other subjects. First, sync_common_subs() is called, then we call this function to finish generating the time tables. Logic wise, it is similar to generate().
14. **free_slots()** - after all required subjects are assigned, we may be left with some empty positions unfilled on the time table. These are filled by any random main subject. It is done in a manner so as to avoid high repetition of one particular main subject.
15. ***substitutions()*** - creates a substitutions file, displaying details of substitution teachers in case a teacher is to be absent.

Note:

Contents of output.pdf and output_substitutions.pdf are given on pages 60 to 90. Please note, this is the final output of the program after generating the time tables and is random every time the program is run, therefore the output attached is just a sample of infinitely many possible outputs.

INSTRUCTIONS
All time tables have 7 rows and 8 columns.
The first row is for Monday, second row for Tuesday and so on till Saturday.
The first column is for the first period, second for second period and so on till eighth period.

6 A
English: Pritha
Maths: Indu Satish
Science: Padma Malini
Sst: Rashmi Ramdas
Hindi: Ashok Kumar Pandey
Tamil: Prema
Computers: Sangeetha V
Computers: Sivagami M
Pt: Esther Rani
Craft: Vijaya Lakshmi
Drawing: Govindarajulu
Yoga: Chandrasekar
Morals: Premabai B
Music: Vidhya Bharathi
Library: Aishwarya J
6A Time Table

Sst	Hindi	Pt	English	Science	Maths	Hindi	Science
Sst	Science	Tamil	English	Maths	Library	Computers	Craft
Hindi	Sst	English	Maths	Tamil	Tamil	Craft	Science
Maths	Science	Hindi	Hindi	Tamil	Sst	English	Pt
CCA	Maths	English	Computers	Science	Sst	Morals	Yoga
Music	English	Drawing	Maths	Science	Tamil	Sst	

6 B
English: Deepa M
Maths: Indu Satish
Science: Padma Balaji
Sst: Nagalakshmi V
Hindi: Neelima Devi
Tamil: Elavarasi P
Computers: Sangeetha V
Computers: Sivagami M
Pt: Esther Rani
Craft: Vrinda Unnikrishnan
Drawing: Govindarajulu
Yoga: Chandrasekar
Morals: Premabai B
Music: Vidhya Bharathi
Library: Aishwarya J
6B Time Table

Tamil	Maths	Sst	Hindi	Tamil	English	Hindi	Science
Maths	Yoga	English	Tamil	Hindi	Science	Sst	Maths
Sst	Pt	English	Science	Maths	Library	Computers	Craft
Morals	English	Science	Tamil	Maths	Hindi	Sst	Craft
CCA	Hindi	Science	Maths	Music	Computers	Sst	English
English	Science	Pt	Drawing	Sst	Tamil	Maths	

6 C
English: Deepa M
Maths: Jamuna N
Science: Padma Malini
Sst: Uma Maheswari
Hindi: Jyoti Sahu
Tamil: Vijayalakshmi Ramesh PK
Computers: Meenakshi Venkatraman
Computers: Sivagami M
Pt: Esther Rani
Craft: Vrinda Unnikrishnan
Drawing: Govindarajulu
Yoga: Chandrasekar
Morals: Premabai B
Music: Vidhya Bharathi
Library: Aishwarya J
6C Time Table

Pt	Science	Computers	Sst	Maths	Hindi	English	Tamil
Science	Tamil	Hindi	Sst	Maths	English	Craft	Hindi
Tamil	Sst	Maths	Science	Pt	Tamil	English	Computers
Tamil	Sst	Science	Maths	English	Craft	Hindi	Drawing
CCA	Maths	Sst	Yoga	English	Science	Library	Morals
Science	Sst	Music	Hindi	Sst	Maths	English	

6 D
English: Deepa M
Maths: Rajalekshmy Balaji
Science: Padma Balaji
Sst: Uma Maheswari
Hindi: Jyoti Sahu
Tamil: Elavarasi P
Computers: Sangeetha V

Computers: Meenakshi Venkatraman
Pt: Esther Rani
Craft: Vrinda Unnikrishnan
Drawing: Govindarajulu
Yoga: Chandrasekar
Morals: Premabai B
Music: Vidhya Bharathi
Library: Aishwarya J

6D Time Table

Hindi	Hindi	Science	English	Maths	Craft	Sst	Tamil
Drawing	Hindi	Science	English	Tamil	Maths	Sst	Yoga
Tamil	Maths	Computers	Sst	English	Craft	Pt	Science
Science	Tamil	Sst	English	Hindi	Maths	Maths	Hindi
CCA	Tamil	Maths	Library	Pt	English	Sst	Science
Science	English	Sst	Maths	Music	Morals	Computers	

7 A
English: Deepa M
Maths: Rajalekshmy Balaji
Science: Padma Malini
Sst: Nagalakshmi V
Hindi: Ashok Kumar Pandey
Tamil: Vijayalakshmi Ramesh PK
Computers: Meenakshi Venkatraman
Computers: Sivagami M
Pt: Esther Rani
Craft: Vijaya Lakshmi
Drawing: Govindarajulu
Yoga: Chandrasekar
Morals: Premabai B
Music: Vidhya Bharathi
Library: Aishwarya J

7A Time Table

Maths	English	Craft	Science	Craft	Tamil	Sst	Hindi
Sst	English	Hindi	Yoga	Maths	Music	Science	English
Computers	Tamil	Pt	Sst	Science	English	Maths	Library
Hindi	Tamil	English	Pt	Maths	Sst	Science	Hindi
CCA	English	Sst	Drawing	Hindi	Maths	Tamil	Science
Computers	Maths	Morals	English	Tamil	Sst	Science	

7 B
English: Pritha
Maths: Indu Satish
Science: Padma Balaji
Sst: Uma Maheswari
Hindi: Jyoti Sahu
Tamil: Elavarasi P
Computers: Meenakshi Venkatraman
Computers: Sivagami M
Pt: Esther Rani
Craft: Vijaya Lakshmi
Drawing: Govindarajulu
Yoga: Chandrasekar
Morals: Premabai B
Music: Vidhya Bharathi
Library: Aishwarya J

7B Time Table

English	Science	Hindi	Library	Computers	Pt	Maths	Sst
Morals	Maths	Sst	Science	Yoga	Tamil	Science	English
Craft	Maths	Science	English	Sst	Computers	Hindi	Drawing
English	Pt	Tamil	Hindi	Science	Maths	Music	Sst
CCA	Science	Hindi	Tamil	Maths	English	Tamil	Sst
Hindi	Tamil	Science	Sst	Craft	Maths	English	

7 C
English: Pritha
Maths: Indu Satish
Science: Padma Malini
Sst: Rashmi Ramdas
Hindi: Jyoti Sahu
Tamil: Prema
Computers: Sivagami M
Computers: Meenakshi Venkatraman
Pt: Esther Rani
Craft: Vijaya Lakshmi
Drawing: Govindarajulu
Yoga: Chandrasekar
Morals: Premabai B
Music: Vidhya Bharathi
Library: Aishwarya J

7C Time Table

Music	Tamil	Science	Maths	Hindi	English	Tamil	Sst
Hindi	English	Craft	Hindi	Science	Sst	Maths	Pt

Tamil	Computers	Science	Drawing	English	Sst	Maths	Hindi
Library	Yoga	Computers	English	Sst	Science	Maths	Craft
CCA	Morals	Sst	Science	Tamil	Maths	Tamil	English
Maths	Sst	English	Science	Pt	Sst	Hindi	

7 D
English: Pritha
Maths: Jamuna N
Science: Ramya R
Sst: Rashmi Ramdas
Hindi: Jyoti Sahu
Tamil: Prema
Computers: Sangeetha V
Computers: Meenakshi Venkatraman
Pt: Esther Rani
Craft: Vijaya Lakshmi
Drawing: Govindarajulu
Yoga: Chandrasekar
Morals: Premabai B
Music: Vidhya Bharathi
Library: Aishwarya J
7D Time Table

Science	Sst	Library	Tamil	English	Maths	Hindi	Pt
Yoga	Tamil	Maths	Craft	Science	English	Hindi	Sst
Morals	Maths	Science	Sst	Craft	Hindi	Science	English
Hindi	Drawing	Sst	Computers	Maths	English	Tamil	Science
CCA	Sst	Tamil	Tamil	Hindi	Maths	English	Science
Science	Pt	Sst	Music	English	Computers	Maths	

8 A
English: Padmini M.S
Maths: Jamuna N
Science: Ramya R
Sst: Nagalakshmi V
Hindi: Ashok Kumar Pandey
Tamil: Elavarasi P
Computers: Sivagami M
Computers: Meenakshi Venkatraman
Pt: Esther Rani
Craft: Vijaya Lakshmi
Drawing: Govindarajulu
Yoga: Chandrasekar
Morals: Premabai B
Music: Vidhya Bharathi
Library: Aishwarya J
8A Time Table

English	Tamil	Morals	Science	English	Hindi	Maths	Sst
Maths	Hindi	Science	Library	Music	English	Hindi	Sst
Pt	Tamil	Sst	Craft	Science	English	Tamil	Maths
Science	Computers	Pt	Sst	English	Hindi	Tamil	Maths
CCA	Yoga	English	Science	Maths	Sst	Computers	Tamil
Hindi	Drawing	Maths	Science	English	Craft	Sst	

8 B
English: Padmini M.S
Maths: Rajalekshmy Balaji
Science: Padma Balaji
Sst: Rashmi Ramdas
Hindi: Jyoti Sahu
Tamil: Prema
Computers: Sangeetha V
Computers: Meenakshi Venkatraman
Pt: Esther Rani
Craft: Vijaya Lakshmi
Drawing: Govindarajulu
Yoga: Chandrasekar
Morals: Premabai B
Music: Vidhya Bharathi
Library: Aishwarya J
8B Time Table

Science	Craft	Music	Sst	Tamil	Maths	English	Hindi
Maths	Science	Sst	Computers	English	Pt	Drawing	Maths
English	Tamil	Hindi	Tamil	Computers	Science	Sst	Maths
Tamil	Maths	Hindi	Science	Yoga	English	Craft	Sst
CCA	Hindi	Pt	Science	Sst	English	Maths	Library
Sst	Hindi	English	Tamil	Science	Maths	Morals	

8 C
English: Pritha
Maths: Jamuna N

Science: Padma Balaji
Sst: Uma Maheswari
Hindi: Ashok Kumar Pandey
Tamil: Vijayalakshmi Ramesh PK
Computers: Meenakshi Venkatraman
Computers: Sangeetha V
Pt: Esther Rani
Craft: Vijaya Lakshmi
Drawing: Govindarajulu
Yoga: Chandrasekar
Morals: Premabai B
Music: Vidhya Bharathi
Library: Aishwarya J

8C Time Table

Sst	Pt	Maths	Tamil	Library	Science	Computers	English
Hindi	Computers	English	Drawing	Pt	Sst	Maths	Science
Science	English	Sst	Tamil	Maths	Morals	Tamil	Music
Maths	English	Tamil	Sst	Hindi	Science	Sst	Yoga
CCA	English	Maths	Sst	Science	Craft	Hindi	Hindi
Sst	Tamil	Craft	Maths	Hindi	English	Science	

8 D
English: Padmini M.S
Maths: Jamuna N
Science: Ramya R
Sst: Nagalakshmi V
Hindi: Ashok Kumar Pandey
Tamil: Prema
Computers: Sangeetha V
Computers: Sivagami M
Pt: Esther Rani
Craft: Vijaya Lakshmi
Drawing: Govindarajulu
Yoga: Chandrasekar
Morals: Premabai B
Music: Vidhya Bharathi
Library: Aishwarya J

8D Time Table

Sst	Maths	Hindi	Hindi	Science	Science	Library	English
Tamil	Sst	Computers	English	Tamil	Maths	Science	Drawing
Science	Morals	English	Maths	Hindi	Craft	Sst	Craft
English	Maths	Music	Tamil	Sst	Pt	Science	Tamil
CCA	Tamil	Hindi	English	Sst	Pt	Science	Maths
Maths	Computers	Science	Sst	Yoga	English	Hindi	

9 A
Sanskrit: Ranganathan R
Tamil: Sayee Lakshmi
Hindi: Elangovan Jain
English: Padmini M.S
Maths: Revathi Sivaram
Science: Flora Bhaskar
Sst: Samundeeswari K
Computers: Sekar K
Pt: Rathinakumar
Craft: Vijaya Lakshmi
Drawing: Govindarajulu
Yoga: Chandrasekar
Morals: Premabai B
Music: Vidhya Bharathi
Library: Aishwarya J

9A Time Table

Craft	Maths	Sst	Language	Music	English	Maths	Science
Sst	English	Pt	Maths	Language	Science	Computers	English
Drawing	Language	Science	Sst	Computers	Maths	English	Yoga
Science	Maths	Language	English	Library	Pt	Sst	Morals
CCA	Practicals	Practicals	Maths	English	Sst	Language	Science
English	Maths	Language	Science	Science	Sst	Craft	

9 B
Sanskrit: Ranganathan R
Tamil: Sayee Lakshmi
Hindi: Elangovan Jain
English: Lavanya Lousia Tonia
Maths: Revathi Sivaram
Science: Kalavathy T
Sst: Samundeeswari K
Computers: Sekar K
Pt: Rathinakumar
Craft: Vijaya Lakshmi
Drawing: Govindarajulu
Yoga: Chandrasekar
Morals: Premabai B
Music: Vidhya Bharathi
Library: Aishwarya J

9B Time Table

Science	Library	Maths	Language	Sst	English	Drawing	Morals
Science	English	Computers	Pt	Language	Craft	Sst	Maths
Sst	Language	Maths	Science	Music	Pt	Sst	English
English	Science	Language	Craft	Maths	Computers	Maths	Sst
CCA	Practicals	Practicals	Science	Maths	English	Language	Sst
Science	English	Language	Sst	Maths	Yoga	English	

9 C
Tamil: Sayee Lakshmi
Hindi: Elangovan Jain
English: Padmini M.S
Maths: Revathi Sivaram
Science: Padma G
Sst: Uma Rameswaran
Computers: Sangeetha V
Computers: Meenakshi Venkatraman
Pt: Rathinakumar
Craft: Vijaya Lakshmi
Drawing: Govindarajulu
Yoga: Chandrasekar
Morals: Premabai B
Music: Vidhya Bharathi
Library: Aishwarya J
9C Time Table

Sst	English	Science	Sst	Maths	Language	Science	Yoga
Maths	Pt	Drawing	Science	Maths	Sst	English	Language
Maths	Pt	Sst	Library	Language	Science	Music	English
Science	Morals	English	Maths	Craft	Computers	Sst	Language
CCA	English	Maths	Language	Practicals	Practicals	Sst	Science
Craft	English	Computers	Maths	Science	Sst	Language	

9 D
Tamil: Sayee Lakshmi
Hindi: Elangovan Jain
English: Lavanya Lousia Tonia
Maths: Revathi Sivaram
Science: Flora Bhaskar
Sst: Uma Rameswaran
Computers: Sangeetha V
Computers: Meenakshi Venkatraman
Pt: Rathinakumar
Craft: Vijaya Lakshmi
Drawing: Govindarajulu
Yoga: Chandrasekar
Morals: Premabai B
Music: Vidhya Bharathi
Library: Aishwarya J
9D Time Table

Computers	English	Sst	Pt	Sst	Language	Science	Maths
English	Maths	Sst	Music	Science	Morals	Craft	Language
Sst	Drawing	Pt	Maths	Language	Yoga	English	Science
Maths	Craft	Science	Sst	English	Library	Computers	Language
CCA	English	Science	Language	Practicals	Practicals	Maths	Sst
Maths	Science	Maths	Sst	English	English	Language	

10 A
Sanskrit: Ranganathan R
Tamil: Sayee Lakshmi
Hindi: Elangovan Jain
English: Lavanya Lousia Tonia
Maths: Uma Sriram
Science: Sreelatha Harish
Sst: Uma Rameswaran
Computers: Sekar K
Pt: Rathinakumar
Craft: Vijaya Lakshmi
Drawing: Govindarajulu
Yoga: Chandrasekar
Morals: Premabai B
Music: Vidhya Bharathi
Library: Aishwarya J
10A Time Table

Language	Practicals	Practicals	Maths	English	Science	Maths	Sst
Craft	Sst	Science	Language	English	Science	Library	Maths
Maths	Computers	Music	Sst	English	Science	Sst	Language
Language	Maths	Computers	Science	Morals	Craft	English	Sst
CCA	Drawing	Science	Sst	Language	Maths	English	Pt
Sst	Pt	English	Maths	Science	Language	Yoga	

10 B
Sanskrit: Ranganathan R
Tamil: Sayee Lakshmi
Hindi: Elangovan Jain
English: Lavanya Lousia Tonia
Maths: Uma Sriram
Science: Sreelatha Harish
Sst: Uma Rameswaran
Computers: Sekar K
Pt: Rathinakumar
Craft: Vijaya Lakshmi
Drawing: Govindarajulu
Yoga: Chandrasekar
Morals: Premabai B
Music: Vidhya Bharathi
Library: Aishwarya J
10B Time Table

Language	Practicals	Practicals	Computers	Science	Maths	Sst	English
Sst	Maths	English	Language	Craft	Drawing	Sst	Science
Music	Sst	Yoga	Science	Maths	English	Pt	Language
Language	Sst	English	Maths	Pt	Morals	Computers	Science
CCA	Science	English	Craft	Language	Sst	Science	Maths
English	Sst	Maths	Science	Library	Language	Maths	

10 C
Tamil: Sayee Lakshmi
Hindi: Elangovan Jain
English: Lavanya Lousia Tonia
Maths: Uma Sriram
Science: Flora Bhaskar
Sst: Samundeeswari K
Computers: Sekar K
Pt: Rathinakumar
Craft: Vijaya Lakshmi
Drawing: Govindarajulu
Yoga: Chandrasekar
Morals: Premabai B
Music: Vidhya Bharathi
Library: Aishwarya J
10C Time Table

Computers	Maths	Language	Practicals	Practicals	Science	English	Sst
Language	Science	Maths	Sst	Pt	English	Yoga	Sst
Science	English	Craft	English	Drawing	Language	Maths	Sst
Maths	Science	Morals	Language	Science	Sst	Library	English
CCA	Music	Maths	Science	Sst	Language	Craft	English
Language	Maths	Science	English	Sst	Computers	Pt	

10 D
Tamil: Sayee Lakshmi
Hindi: Elangovan Jain
English: Somini Sudeep
Maths: Uma Sriram
Science: Flora Bhaskar
Sst: Samundeeswari K
Computers: Sekar K
Pt: Rathinakumar
Craft: Vijaya Lakshmi
Drawing: Govindarajulu
Yoga: Chandrasekar
Morals: Premabai B
Music: Vidhya Bharathi
Library: Aishwarya J
10D Time Table

Science	Sst	Language	Practicals	Practicals	Craft	English	Maths
Language	Sst	Morals	Science	Maths	English	Pt	Computers
English	Sst	English	Science	Science	Language	Yoga	Maths
English	Music	Sst	Language	Sst	Drawing	Science	Maths
CCA	Science	Computers	Sst	English	Language	Maths	Craft
Language	Sst	Library	English	Pt	Maths	Science	

11 A
Physics: Kalavathy T
Chemistry: Padma G
English: Anupama
Maths: Hemalatha P
Computers: Sekar K
Pt: Rathinakumar
Library: Aishwarya J
11A Time Table

Chemistry	Computers	English	English	Maths	Physics	Maths	Pt
Chemistry	Maths	English	Computers	P/C Lab	P/C Lab	Physics	Physics

English	Maths	Computers	Chemistry	Physics	Computers	Chemistry	Computers
English	English	P/C Lab	P/C Lab	Computers	Maths	Chemistry	Physics
CCA	Computers	Library	Chemistry	Pt	Maths	English	Physics
Chemistry	Computers	Physics	Maths	English	WE	WE	

11 B
Physics: Vijaya Devanathan
Chemistry: Sreelatha Harish
Biology: Vasanthy
English: Somini Sudeep
Maths: Hemalatha P
Pt: Rathinakumar
Library: Aishwarya J
11B Time Table

Chemistry	Biology	Maths	English	Physics	English	P/C Lab	P/C Lab
P/C Lab	P/C Lab	Physics	Maths	Chemistry	Pt	Biology	English
Library	Chemistry	Biology	Pt	Maths	Biology	Physics	English
Bio Lab	Bio Lab	Chemistry	Biology	Chemistry	Physics	English	Maths
CCA	Maths	English	Biology	Chemistry	Physics	Maths	Physics
Physics	Chemistry	English	Biology	Maths	WE	WE	

11 C
Accounts: Bhuvaneswari R
Business: Anbazhagan K
English: Anupama
Maths: Hemalatha P
Economics: Usha Nandhini
Pt: Rathinakumar
Library: Aishwarya J
11C Time Table

Maths	Business	Accounts	Economics	Accounts	English	Pt	Library
Maths	Business	Economics	Economics	English	Maths	English	Accounts
Pt	Accounts	Business	Business	English	Maths	English	Economics
Maths	Maths	English	Economics	Business	Business	Accounts	Economics
CCA	Accounts	Accounts	Accounts	Maths	English	Business	Economics
English	Business	Maths	Accounts	Economics	WE	WE	

11 D
Accounts: Bhuvaneswari R
Business: Sowmya Swamy
Entre: Sowmya Swamy
English: Somini Sudeep
Economics: Malabika Ghosh
Pt: Rathinakumar
Library: Aishwarya J
11D Time Table

English	Entre	Economics	Economics	Business	Accounts	Economics	Accounts
Library	English	Business	Economics	English	Accounts	Entre	Pt
Business	English	Entre	Accounts	English	Entre	Economics	Business
Entre	Entre	English	Entre	Accounts	English	Economics	Business
CCA	English	Pt	Entre	Business	English	Accounts	Economics
Accounts	English	Economics	Entre	Business	WE	WE	

12 A
Physics: Vijaya Devanathan
Chemistry: Padma G
English: Anupama
Maths: Hemalatha P
Computers: Sekar K
Pt: Rathinakumar
Library: Aishwarya J
12A Time Table

English	Chemistry	Physics	Maths	Computers	Physics	Computers	Maths
Computers	Chemistry	Chemistry	Physics	Maths	English	P/C Lab	P/C Lab
P/C Lab	P/C Lab	Library	Maths	Chemistry	Physics	Computers	English
Computers	Pt	Maths	Physics	Chemistry	English	Physics	Pt
CCA	English	Maths	Physics	Computers	Computers	Chemistry	English
Maths	English	Chemistry	Physics	Computers	WE	WE	

12 B
Physics: Kalavathy T
Chemistry: Sreelatha Harish
Biology: Vasanthy

English: Anupama
Maths: Geetha Ganesh
Pt: Rathinakumar
Library: Aishwarya J
12B Time Table

Maths	English	Biology	Chemistry	English	Bio Lab	Bio Lab	Physics
Pt	Biology	Physics	Chemistry	Maths	Maths	Chemistry	English
Chemistry	Maths	Physics	English	Biology	Physics	P/C Lab	P/C Lab
Physics	Chemistry	Pt	English	Biology	P/C Lab	P/C Lab	Maths
CCA	Library	Maths	English	English	Biology	Physics	Chemistry
Biology	Physics	Chemistry	English	Maths	WE	WE	

12 C
Accounts: Anbazhagan K
Business: Sowmya Swamy
English: Somini Sudeep
Maths: Geetha Ganesh
Economics: Usha Nandhini
Pt: Rathinakumar
Library: Aishwarya J
12C Time Table

Business	Economics	English	Maths	Pt	Accounts	Maths	English
Business	Maths	English	Business	Economics	Economics	Accounts	Accounts
Economics	Business	Maths	Maths	Library	Accounts	English	Accounts
Maths	Accounts	Economics	English	Economics	Business	Business	English
CCA	Economics	Business	Economics	Accounts	Maths	Pt	English
English	Maths	Business	Economics	Accounts	WE	WE	

12 D
Accounts: Anbazhagan K
Business: Sowmya Swamy
Entre: Sowmya Swamy
English: Somini Sudeep
Economics: Malabika Ghosh
Pt: Rathinakumar
Library: Aishwarya J
12D Time Table

Accounts	English	Entre	Business	Economics	Entre	Entre	Economics
Accounts	Entre	Economics	English	Business	Accounts	English	Business
Accounts	Economics	Economics	English	Entre	English	Business	Pt
Accounts	English	Business	Accounts	Entre	Economics	Pt	Library
CCA	Business	Economics	Accounts	Economics	Entre	English	Accounts
Entre	Business	Accounts	Economics	English	WE	WE	

Physics Lab

	10A	10B	10C	10D		11B	11B
11B	11B			11A	11A	12A	12A
12A	12A					12B	12B
		11A	11A		12B	12B	
	9A	9B		9C	9D		

Chemistry Lab

	10A	10B	10C	10D		11B	11B
11B	11B			11A	11A	12A	12A
12A	12A					12B	12B
		11A	11A		12B	12B	
	9A	9B		9C	9D		

Biology Lab

	10A	10B	10C	10D	12B	12B	
11B	11B						
	9A	9B		9C	9D		

Computer Lab Junior

9D		6C		7B		8C	
	8C	8D	8B			6A	
7A	7C	6D		8B	7B	6B	6C
	8A	7C	7D		9C	9D	
			6A		6B	8A	
7A	8D	9C			7D	6D	

Computer Lab Senior

10C			10B				
		9B				9A	10D
	10A			9A			
		10A			9B	10B	
		10D					
					10C		

Somini Sudeep

11D	12D	12C	11B		11B	10D	12C
	11D	12C	12D	11D	10D	12D	11B
10D	11D	10D	12D	11D	12D	12C	11B
10D	12D		12C		11D	11B	12C
	11D	11B		10D	11D	12D	12C
12C	11D	11B	10D	12D			

Anupama

12A	12B	11A	11A	12B	11C		
		11A		11C	12A	11C	12B
11A			12B	11C		11C	12A
11A	11A	11C	12B		12A		
	12A			12B	11C	11A	12A
11C	12A		12B	11A			

Lavanya Lousia Tonia

	9D			10A	9B	10C	10B
9D	9B	10B		10A	10C		
	10C		10C	10A	10B	9D	9B
9B		10B		9D		10A	10C
	9D	10B			9B	10A	10C
10B	9B	10A	10C	9D	9D	9B	

Padmini M.S

8A	9C			8A	9A	8B	8D
	9A		8D	8B	8A	9C	9A
8B		8D			8A	9A	9C
8D		9C	9A	8A	8B		
	9C	8A	8D	9A	8B		
9A	9C	8B		8A	8D		

Pritha

7B			6A	7D	7C		8C
	7C	8C	6A		7D		7B
	8C	6A	7B	7C			7D
7B	8C		7C		7D	6A	
	8C	6A			7B	7D	7C
	6A	7C		7D	8C	7B	

Deepa M

	7A		6D		6B	6C	
	7A	6B	6D		6C		7A
		6B		6D	7A	6C	
	6B	7A	6D	6C			
	7A			6C	6D		6B
6B	6D		7A			6C	

Gunjan Agarwal
Schedule is empty.

Elangovan Jain

10A		10C	9A		9C		
10C			10A	9A			9C
	9A			9C	10C		10A
10A		9A	10C				9C
			9C	10A	10C	9A	
10C		9A			10A	9C	

Ashok Kumar Pandey

	6A	8D	8D		8A	6A	7A
8C	8A	7A				8A	
6A				8D			
7A		6A	6A	8C	8A		7A
		8D		7A		8C	8C
8A				8C		8D	

Jyoti Sahu

6D	6D	7B		7C	6C	7D	8B
7C	6D	6C	7C			7D	6C
		8B			7D	7B	7C
7D		8B	7B	6D		6C	6D
	8B	7B		7D			
7B	8B		6C			7C	

Neelima Devi

			6B			6B	
				6B			
					6B		
	6B						

Sayee Lakshmi

10A		10C	9A		9C		
10C			10A	9A			9C
	9A			9C	10C		10A
10A		9A	10C				9C
			9C	10A	10C	9A	
10C		9A			10A	9C	

Elavarasi P

6B	8A			6B			6D
			6B	6D	7B		
6D	8A					8A	
	6D	7B	6B			8A	
	6D		7B			7B	8A
	7B				6B		

Prema

	7C		7D	8B		7C		
8D	7D	6A		8D				
7C	8B		8B	6A	6A			
8B			8D	6A		7D	8D	
	8D	7D	7D	7C		7C		
			8B		6A			

Vijayalakshmi Ramesh PK

			8C		7A		6C	
	6C							
6C	7A		8C		6C	8C		
6C	7A	8C						
						7A		
	8C			7A				

Ranganathan R

10A			9A					
			10A	9A				
	9A						10A	
10A		9A						
				10A		9A		
		9A			10A			

Geetha Ganesh

12B			12C			12C		
	12C			12B	12B			
	12B	12C	12C					
12C							12B	
		12B			12C			
	12C			12B				

Hemalatha P

11C		11B	12A	11A		11A	12A	
11C	11A		11B	12A	11C			
	11A		12A	11B	11C			
11C	11C	12A			11A		11B	
	11B	12A		11C	11A	11B		
12A		11C	11A	11B				

Uma Sriram

	10C		10A		10B	10A	10D	
	10B	10C		10D			10A	
10A				10B		10C	10D	
10C	10A		10B				10D	
		10C			10A	10D	10B	
	10C	10B	10A		10D	10B		

Revathi Sivaram

	9A	9B		9C		9A	9D	
9C	9D		9A	9C			9B	
9C		9B	9D		9A			
9D	9A		9C	9B		9B		
		9C	9A	9B		9D		
9D	9A	9D	9C	9B				

Jamuna N

	8D	8C		6C	7D	8A	
8A		7D		6C	8D	8C	
	7D	6C	8D	8C			8A
8C	8D		6C	7D			8A
	6C	8C		8A	7D		8D
8D		8A	8C		6C	7D	

Indu Satish

	6B		7C		6A	7B	
6B	7B			6A		7C	6B
	7B		6A	6B		7C	
6A				6B	7B	7C	
	6A		6B	7B	7C		
7C			6A		7B	6B	

Rajalekshmy Balaji

7A				6D	8B		
8B				7A	6D		8B
	6D					7A	8B
	8B			7A	6D	6D	
		6D			7A	8B	
	7A		6D		8B		

Sreelatha Harish

11B	10A	10B	12B	10B	10A	11B	11B
11B	11B	10A	12B	11B	10A	12B	10B
12B	11B		10B		10A	12B	12B
	12B	11B	10A	11B	12B	12B	10B
	10B	10A	12B	11B		10B	12B
	11B	12B	10B	10A			

Padma G

11A	12A	9C	10C	10D		9C	
11A	12A	12A	9C	11A	11A	12A	12A
12A	12A		11A	12A	9C	11A	
9C		11A	11A	12A		11A	
	9A	9B	11A	9C	9D	12A	9C
11A		12A		9C			

Vasanthy

	11B	12B	10C	10D	12B	12B	
	12B					11B	
		11B		12B	11B		
11B	11B		11B	12B			
	9A	9B	11B		12B		
12B			11B				

Flora Bhaskar

10D	10A	10B			10C	9D	9A
	10C		10D	9D	9A		
10C		9A	10D	10D			9D
9A	10C	9D		10C		10D	
	10D	9D	10C	9C	9D		9A
	9D	10C	9A	9A		10D	

Vijaya Devanathan

		12A		11B	12A	11B	11B
11B	11B	11B	12A			12A	12A
12A	12A				12A	11B	
					11B	12A	
			12A		11B		11B
11B			12A				

Kalavathy T

9B	10A	10B	10C	10D	11A		12B
9B		12B		11A	11A	11A	11A
		12B	9B	11A	12B	12B	12B
12B	9B	11A	11A		12B	12B	11A
	9A	9B	9B	9C	9D	12B	11A
9B	12B	11A					

Padma Balaji

8B	7B	6D			8C		6B
	8B	6D	7B		6B	7B	8C
8C		7B	6B		8B		6D
6D		6B	8B	7B	8C		
	7B	6B	8B	8C			6D
6D	6B	7B		8B		8C	

Ramya R

7D			8A	8D	8D		
		8A		7D		8D	
8D		7D		8A		7D	
8A						8D	7D
			8A			8D	7D
7D		8D	8A				

Padma Malini

	6C	7C	7A	6A			6A
6C	6A			7C		7A	
		7C	6C	7A			6A
	6A	6C			7C	7A	
			7C	6A	6C		7A
6C			7C	6A		7A	

Usha Nandhini

	12C		11C				
		11C	11C	12C	12C		
12C							11C
		12C	11C	12C			11C
	12C		12C				11C
			12C	11C			

Malabika Ghosh

		11D	11D	12D		11D	12D
		12D	11D				
	12D	12D				11D	
12D					12D	11D	
				12D			11D
		11D	12D				

Sowmya Swamy

12C	11D	12D	12D	11D	12D	12D	
12C	12D	11D	12C	12D		11D	12D
11D	12C	11D		12D	11D	12D	11D
11D	11D	12D	11D	12D	12C	12C	11D
	12D	12C	11D	11D	12D		
12D	12D	12C	11D	11D			

Anbazhagan K

12D	11C				12C		
12D	11C				12D	12C	12C
12D		11C	11C		12C		12C
	12C		12D	11C	11C		
		12D	12D	12C		11C	12D
	11C	12D		12C			

Bhuvaneswari R

		11C		11C	11D		11D
					11D		11C
	11C		11D				
		11D		11D		11C	
	11C	11C	11C			11D	
11D			11C				

Sekar K

10C	11A		10B	12A		12A	
12A		9B	11A			9A	10D
	10A	11A		9A	11A	12A	11A
12A		10A	12A	11A	9B	10B	
	11A	10D		12A	12A		
	11A			12A	10C		

Sangeetha V

9D						8C	
	8C	8D	8B			6A	
		6D		8B		6B	
			7D		9C	9D	
			6A		6B		
	8D	9C			7D	6D	

Meenakshi Venkatraman

9D		6C		7B		8C	
	8C		8B				
7A	7C	6D		8B	7B		6C
	8A	7C	7D		9C	9D	
						8A	
7A		9C			7D	6D	

Sivagami M

		6C		7B			
		8D				6A	
7A	7C				7B	6B	6C
	8A	7C					
			6A		6B	8A	
7A	8D						

Samundeeswari K

	10D	9A		9B			10C

9A	10D		10C			9B	10C
9B	10D		9A			9B	10C
		10D		10D	10C	9A	9B
			10D	10C	9A		9B
	10D		9B	10C	9A		

Uma Rameswaran

9C		9D	9C	9D		10B	10A
10B	10A	9D			9C	10B	
9D	10B	9C	10A			10A	
	10B		9D			9C	10A
			10A		10B	9C	9D
10A	10B		9D		9C		

Rashmi Ramdas

6A	7D		8B				7C
6A		8B			7C		7D
	6A		7D		7C	8B	
		7D		7C	6A		8B
	7D	7C		8B	6A		
8B	7C	7D			7C	6A	

Uma Maheswari

8C			6C			6D	7B
		7B	6C		8C	6D	
	6C	8C	6D	7B			
	6C	6D	8C			8C	7B
		6C	8C			6D	7B
8C	6C	6D	7B	6C			

Nagalakshmi V

8D		6B				7A	8A
7A	8D					6B	8A
6B		8A	7A			8D	
			8A	8D	7A	6B	
		7A		8D	8A	6B	
			8D	6B	7A	8A	

Rathinakumar

			9D	12C		11C	11A
12B	9C	9A	9B	10C	11B	10D	11D
11C	9C	9D	11B		9B	10B	12D
	12A	12B		10B	9A	12D	12A
		11D		11A		12C	10A
	10A			10D		10C	

Esther Rani

6C	8C	6A			7B		7D
				8C	8B		7C
8A	6B	7A		6C		6D	
	7B	8A	7A		8D		6A
		8B		6D	8D		
	7D	6B		7C			

Vijaya Lakshmi

9A	8B	7A		7A	10D		

10A		7C	7D	10B	9B	9D	6A
7B		10C	8A	7D	8D	6A	8D
	9D		9B	9C	10A	8B	7C
			10B		8C	10C	10D
9C		8C		7B	8A	9A	

Govindarajulu

						9B	
6D		9C	8C		10B	8B	8D
9A	9D		7C	10C			7B
	7D				10D		6C
	10A		7A				
	8A	6A	6B				

Chandrasekar

							9C
7D	6B		7A	7B		10C	6D
		10B			9D	10D	9A
	7C			8B			8C
	8A		6C				6A
				8D	9B	10A	

Premabai B

		8A					9B
7B		10D			9D		
7D	8D				8C		
6B	9C	10C		10A	10B		9A
	7C					6A	6C
		7A			6D	8B	

Vidhya Bharathi

7C		8B		9A			
			9D	8A	7A		
10B		10A		9B		9C	8C
	10D	8D				7B	
	10C			6B			
6A		6C	7D	6D			

Vrinda Unnikrishnan

					6D		
						6C	
					6D		6B
					6C		6B

Aishwarya J

	9B	7D	7B	8C		8D	11C
11D			8A		6A	10A	
11B		12A	9C	12C	6B		7A
7C				9A	9D	10C	12D
	12B	11A	6D			6C	8B
		10D		10B			

INSTRUCTIONS FOR SUBSTITUTION TEACHERS
IF A TEACHER IS TO BE ABSENT,
then the table shows the IDs of all available teachers.

For example, if it shows
7A: 23, 25, 21
it means you can select any one of these free teachers,
and assign them to class 7A at that particular period.

The teachers are ORDERED from HIGHEST PRIORITY to LOWEST PRIORITY
PRIORITY is given to teachers who already teach the class
This ensures that teaching can continue even during substitutions

S.No.	Name	Subject	Grades	Min classes	Max classes
1.	Somini Sudeep	English	10 11 12	4	5
2.	Anupama	English	11 12	4	5
3.	Lavanya Lousia Tonia	English	9 10	4	5
4.	Padmini M.S	English	8 9	4	5
5.	Pritha	English	6 7 8	4	5
6.	Deepa M	English	6 7	4	5
7.	Gunjan Agarwal	English	4 5	4	5
8.	Elangovan Jain	Hindi	8 9 10	4	7
9.	Ashok Kumar Pandey	Hindi	5 6 7 8	4	7
10.	Jyoti Sahu	Hindi	5 6 7 8	4	7
11.	Neelima Devi	Hindi	4 5 6	5	7
12.	Sayee Lakshmi	Tamil	9 10	4	6
13.	Elavarasi P	Tamil	6 7 8	4	5
14.	Prema	Tamil	6 7 8	4	5
15.	Vijayalakshmi Ramesh PK	Tamil	6 7 8	4	5
16.	Ranganathan R	Sanskrit	9 10	4	6
17.	Geetha Ganesh	Maths	12	2	2
18.	Hemalatha P	Maths	11 12	4	5
19.	Uma Sriram	Maths	10	4	5
20.	Revathi Sivaram	Maths	9	4	5
21.	Jamuna N	Maths	6 7 8	4	5
22.	Indu Satish	Maths	6 7	4	5
23.	Rajalekshmy Balaji	Maths	6 7 8	4	4
24.	Sreelatha Harish	Science	10 11 12	3	4
25.	Padma G	Science	9 11 12	3	4
26.	Vasanthy	Science	11 12	2	2
27.	Flora Bhaskar	Science	9 10	2	4
28.	Vijaya Devanathan	Science	11 12	2	2
29.	Kalavathy T	Science	9 11 12	3	4
30.	Padma Balaji	Science	6 7 8	3	5
31.	Ramya R	Science	7 8	3	5
32.	Padma Malini	Science	6 7	4	5
33.	Usha Nandhini	Economics	11 12	2	2
34.	Malabika Ghosh	Economics	11 12	3	4
35.	Sowmya Swamy	Commerce	11 12	2	5
36.	Anbazhagan K	Commerce	11 12	2	5
37.	Bhuvaneswari R	Commerce	11 12	2	5
38.	Sekar K	Computers	9 10 11 12	4	5
39.	Sangeetha V	Computers	6 7 8 9	14	17
40.	Meenakshi Venkatraman	Computers	6 7 8 9	9	17
41.	Sivagami M	Computers	6 7 8	8	10
42.	Samundeeswari K	Sst	9 10	4	5
43.	Uma Rameswaran	Sst	9 10	4	5
44.	Rashmi Ramdas	Sst	6 7 8	4	5
45.	Uma Maheswari	Sst	6 7 8	3	4
46.	Nagalakshmi V	Sst	6 7 8	4	4
47.	Rathinakumar	Pt	9 10 11 12	16	22
48.	Esther Rani	Pt	6 7 8	11	15
49.	Vijaya Lakshmi	Craft	6 7 8 9 10 11 12	13	30
51.	Govindarajulu	Drawing	4 5 6 7 8 9 10	20	30
52.	Chandrasekar	Yoga	4 5 6 7 8 9 10	20	30
53.	Premabai B	Morals	6 7 8 9 10	20	25
54.	Vidhya Bharathi	Music	6 7 8 9 10 11 12	20	35
55.	Vrinda Unnikrishnan	Craft	4 5 6	6	10
50.	Aishwarya J	Library	6 7 8 9 10 11 12	20	30

SUBSTITUTION FOR TEACHER Somini Sudeep

11D: 34, 50, 47, 37	12D: 34, 47, 5, 7	12C: 33, 17, 47, 36	11B: 28, 3, 4, 7		11B: 18, 47, 50, 7	10D: 42, 12, 52, 54	12C: 33, 17, 35, 36
	11D: 34, 37, 50, 2	12C: 17, 36, 50, 4	12D: 36, 2, 3, 7	11D: 34, 37, 50, 5	10D: 42, 19, 38, 12	12D: 34, 3, 5, 6	11B: 18, 50, 26, 3
10D: 12, 8, 52, 38	11D: 50, 2, 4, 6	10D: 42, 19, 51, 53	12D: 34, 35, 35, 4	11D: 34, 37, 47, 4	12D: 34, 2, 5, 7	12C: 17, 33, 36, 50	11B: 18, 28, 26, 6
10D: 42, 47, 49, 54	12D: 34, 50, 3, 4		12C: 17, 50, 47, 3		11D: 37, 3, 6, 7	11B: 18, 26, 2, 4	12C: 36, 2, 4, 5
	11D: 34, 47, 7, 8	11B: 28, 2, 6, 7		10D: 19, 51, 53, 50	11D: 34, 47, 50, 37	12D: 34, 35, 35, 4	12C: 17, 35, 4, 7
12C: 17, 33, 47, 50	11D: 34, 50, 37, 7	11B: 26, 47, 28, 2	10D: 52, 49, 12, 38	12D: 34, 6, 7, 8			

SUBSTITUTION FOR TEACHER Anupama

12A: 28, 50, 47, 3	12B: 17, 47, 5, 7	11A: 38, 47, 3, 4	11A: 3, 4, 7, 10	12B: 17, 1, 6, 7	11C: 33, 18, 47, 50		
		11A: 18, 50, 4, 7		11C: 36, 37, 50, 5	12A: 38, 28, 7, 8	11C: 33, 18, 37, 3	12B: 17, 50, 26, 3
11A: 18, 38, 29, 3			12B: 26, 4, 6, 7	11C: 33, 36, 37, 47		11C: 33, 18, 37, 50	12A: 18, 28, 25, 6
11A: 47, 6, 7, 11	11A: 38, 25, 50, 3	11C: 36, 50, 1, 5	12B: 17, 47, 50, 3		12A: 25, 3, 6, 7		
	12A: 47, 28, 7, 8			12B: 17, 26, 50, 3	11C: 33, 50, 37, 36	11A: 38, 4, 6, 7	12A: 18, 38, 4, 7
11C: 33, 47, 50, 36	12A: 18, 25, 50, 28		12B: 17, 29, 47, 50	11A: 29, 6, 7, 8			

SUBSTITUTION FOR TEACHER Lavanya Lousia Tonia

	9D: 43, 54, 52, 53			10A: 19, 53, 8, 12	9B: 42, 20, 38, 54	10C: 42, 53, 54, 12	10B: 12, 51, 16, 49
9D: 27, 54, 39, 40	9B: 29, 38, 8, 49	10B: 50, 54, 8, 16		10A: 43, 53, 38, 50	10C: 19, 42, 12, 38		
	10C: 27, 19, 52, 50		10C: 19, 54, 12, 38	10A: 43, 24, 16, 53	10B: 43, 19, 51, 16	9D: 27, 20, 40, 53	9B: 20, 53, 6, 7
9B: 42, 54, 51, 52		10B: 43, 19, 52, 50		9D: 43, 12, 51, 8		10A: 19, 53, 8, 16	10C: 27, 54, 38, 2
	9D: 43, 20, 49, 47	10B: 43, 52, 53, 16			9B: 20, 53, 16, 54	10A: 38, 52, 51, 54	10C: 51, 54, 8, 12
10B: 19, 24, 38, 52	9B: 8, 53, 49, 12	10A: 43, 38, 52, 47	10C: 38, 52, 47, 49	9D: 43, 51, 53, 39	9D: 20, 27, 47, 50	9B: 20, 29, 42, 50	

SUBSTITUTION FOR TEACHER Padmini M.S

8A: 21, 41, 9, 53	9C: 43, 53, 12, 39			8A: 46, 52, 48, 51	9A: 20, 42, 52, 47	8B: 30, 44, 23, 54	8D: 31, 21, 51, 14
	9A: 38, 8, 50, 16		8D: 31, 21, 46, 53	8B: 44, 30, 50, 10	8A: 46, 31, 9, 52	9C: 20, 53, 8, 40	9A: 27, 53, 50, 16
8B: 44, 23, 39, 52		8D: 41, 14, 51, 53			8A: 21, 31, 46, 13	9A: 20, 27, 51, 12	9C: 43, 20, 25, 53
8D: 46, 39, 41, 52		9C: 43, 20, 39, 52	9A: 27, 42, 52, 53	8A: 31, 48, 54, 51	8B: 54, 14, 10, 52		
	9C: 20, 43, 39, 40	8A: 31, 41, 40, 53	8D: 46, 21, 54, 9	9A: 52, 49, 50, 53	8B: 30, 52, 14, 54		
9A: 42, 27, 50, 38	9C: 25, 54, 8, 50	8B: 23, 52, 10, 14		8A: 21, 31, 53, 41	8D: 31, 50, 41, 48		

SUBSTITUTION FOR TEACHER Pritha

7B: 22, 52, 53, 41			6A: 39, 41, 52, 53	7D: 44, 52, 53, 51	7C: 44, 32, 53, 54		8C: 21, 40, 39, 49
	7C: 44, 41, 54, 53	8C: 50, 15, 52, 48	6A: 22, 44, 32, 14		7D: 31, 10, 52, 39		7B: 45, 53, 50, 54
	8C: 30, 54, 52, 9	6A: 22, 44, 53, 41	7B: 48, 54, 13, 53	7C: 44, 41, 52, 10			7D: 31, 44, 53, 14
7B: 45, 49, 40, 41	8C: 30, 50, 39, 9		7C: 22, 44, 32, 50		7D: 21, 31, 52, 10	6A: 44, 51, 52, 9	
	8C: 45, 49, 9, 15	6A: 32, 22, 53, 52			7B: 45, 30, 51, 50	7D: 21, 44, 51, 52	7C: 22, 44, 51, 48
	6A: 22, 32, 53, 9	7C: 22, 32, 14, 41		7D: 21, 31, 44, 51	8C: 30, 45, 48, 50	7B: 45, 54, 50, 48	

SUBSTITUTION FOR TEACHER Deepa M

	7A: 23, 46, 53, 40		6D: 23, 30, 10, 51		6B: 46, 11, 54, 52	6C: 32, 53, 41, 48	
	7A: 23, 51, 49, 54	6B: 22, 46, 55, 54	6D: 23, 55, 53, 48		6C: 32, 15, 10, 55		7A: 32, 15, 53, 9
		6B: 22, 53, 41, 51		6D: 23, 30, 52, 55	7A: 23, 46, 32, 48	6C: 21, 32, 45, 51	
	6B: 22, 46, 30, 50	7A: 23, 46, 51, 49	6D: 23, 55, 51, 50	6C: 45, 32, 40, 41			
	7A: 23, 32, 46, 48			6C: 45, 40, 41, 53	6D: 45, 30, 53, 10		6B: 22, 46, 48, 51
6B: 46, 53, 51, 52	6D: 53, 50, 52, 55		7A: 9, 48, 15, 40			6C: 45, 51, 55, 48	

SUBSTITUTION FOR TEACHER Gunjan Agarwal

SUBSTITUTION FOR TEACHER Elangovan Jain

10A: 3, 19, 53, 52		10C: 3, 19, 51, 47	9A: 42, 4, 20, 27		9C: 43, 25, 20, 47		
10C: 19, 27, 54, 1			10A: 43, 19, 3, 53	9A: 42, 38, 51, 50			9C: 43, 53, 50, 39
	9A: 20, 4, 27, 52			9C: 43, 20, 4, 52	10C: 27, 42, 19, 54		10A: 43, 53, 6, 7
10A: 24, 43, 49, 51		9A: 20, 50, 52, 51	10C: 42, 3, 27, 47				9C: 20, 25, 4, 54
			9C: 40, 53, 47, 54	10A: 3, 43, 19, 51	10C: 51, 53, 54, 47	9A: 4, 27, 42, 38	
10C: 27, 19, 42, 50		9A: 42, 52, 38, 47			10A: 24, 54, 50, 51	9C: 20, 4, 25, 43	

SUBSTITUTION FOR TEACHER Ashok Kumar Pandey

	6A: 5, 53, 51, 52	8D: 31, 4, 51, 14	8D: 4, 21, 46, 51		8A: 46, 53, 52, 13	6A: 32, 5, 44, 48	7A: 6, 23, 49, 40
8C: 45, 30, 5, 15	8A: 31, 21, 51, 53	7A: 46, 32, 23, 54				8A: 54, 48, 13, 40	
6A: 32, 44, 22, 5				8D: 46, 4, 41, 53			
7A: 23, 32, 6, 46		6A: 5, 22, 50, 39	6A: 22, 32, 44, 54	8C: 45, 5, 48, 54	8A: 31, 21, 41, 54		7A: 23, 6, 32, 46
		8D: 31, 52, 49, 51		7A: 23, 15, 40, 41		8C: 21, 30, 39, 52	8C: 48, 51, 15, 40
8A: 46, 50, 51, 52				8C: 21, 51, 40, 39		8D: 31, 4, 41, 14	

SUBSTITUTION FOR TEACHER Jyoti Sahu

6D: 6, 51, 52, 55	6D: 23, 45, 53, 39	7B: 22, 5, 45, 51		7C: 22, 44, 52, 53	6C: 32, 45, 53, 51	7D: 31, 5, 44, 53	8B: 23, 49, 54, 51
7C: 5, 48, 54, 41	6D: 23, 45, 51, 50	6C: 32, 50, 55, 15	7C: 44, 32, 22, 41			7D: 44, 5, 54, 14	6C: 21, 45, 32, 53
		8B: 44, 23, 51, 53			7D: 5, 21, 31, 48	7B: 5, 30, 45, 51	7C: 22, 44, 53, 14
7D: 44, 52, 51, 39		8B: 23, 50, 51, 39	7B: 22, 51, 50, 41	6D: 45, 55, 13, 48		6C: 6, 21, 48, 15	6D: 30, 23, 6, 13
	8B: 23, 39, 40, 48	7B: 22, 49, 53, 51		7D: 5, 31, 51, 49			
7B: 5, 50, 48, 13	8B: 14, 54, 49, 40		6C: 52, 48, 15, 50			7C: 54, 51, 48, 14	

SUBSTITUTION FOR TEACHER Neelima Devi

			6B: 46, 30, 39, 41			6B: 30, 41, 13, 48	
				6B: 46, 30, 6, 51			
					6B: 6, 54, 52, 41		
	6B: 46, 48, 55, 39						

SUBSTITUTION FOR TEACHER Sayee Lakshmi

10A: 19, 3, 50, 52		10C: 19, 3, 51, 38	9A: 27, 42, 4, 20		9C: 20, 25, 43, 50		
10C: 27, 19, 54, 1			10A: 19, 3, 43, 53	9A: 42, 50, 53, 51			9C: 43, 54, 50, 39
	9A: 27, 4, 20, 52			9C: 4, 43, 20, 52	10C: 27, 19, 42, 54		10A: 43, 53, 6, 7
10A: 24, 43, 54, 47		9A: 20, 49, 51, 50	10C: 42, 3, 27, 53				9C: 25, 4, 20, 39
			9C: 47, 40, 54, 53	10A: 3, 43, 19, 51	10C: 47, 51, 50, 54	9A: 42, 27, 4, 38	
10C: 42, 19, 27, 52		9A: 42, 38, 52, 47			10A: 24, 47, 51, 50	9C: 4, 25, 43, 20	

SUBSTITUTION FOR TEACHER Elavarasi P

6B: 6, 22, 11, 41	8A: 31, 46, 51, 54			6B: 22, 46, 6, 30		6D: 23, 6, 54, 51
			6B: 46, 22, 53, 11	6D: 45, 6, 30, 10	7B: 22, 10, 52, 40	
6D: 23, 45, 6, 52	8A: 4, 31, 46, 9				8A: 21, 9, 51, 40	
	6D: 30, 55, 50, 10	7B: 5, 22, 52, 49	6B: 22, 53, 51, 11		8A: 4, 21, 52, 51	
	6D: 23, 45, 39, 40		7B: 5, 54, 40, 10		7B: 30, 22, 54, 52	8A: 4, 46, 41, 40
	7B: 22, 52, 53, 49				6B: 30, 6, 11, 50	

SUBSTITUTION FOR TEACHER Prema

	7C: 5, 54, 53, 52		7D: 21, 53, 39, 40	8B: 30, 44, 51, 48		7C: 5, 32, 44, 48	
8D: 31, 4, 48, 54	7D: 44, 31, 21, 50	6A: 22, 32, 54, 52		8D: 46, 50, 53, 51			
7C: 22, 44, 32, 5	8B: 30, 4, 49, 39		8B: 23, 4, 52, 54	6A: 44, 53, 41, 52	6A: 32, 22, 5, 39		
8B: 44, 23, 52, 51			8D: 31, 50, 53, 52	6A: 32, 5, 48, 39		7D: 21, 44, 52, 51	8D: 4, 46, 39, 41
	8D: 46, 31, 48, 39	7D: 31, 49, 52, 54	7D: 5, 21, 44, 48	7C: 5, 41, 40, 53		7C: 32, 44, 22, 48	
			8B: 4, 30, 44, 53		6A: 32, 9, 51, 41		

SUBSTITUTION FOR TEACHER Vijayalakshmi Ramesh PK

			8C: 21, 30, 40, 39		7A: 32, 46, 51, 52	6C: 21, 6, 54, 40
	6C: 21, 45, 50, 51					
6C: 6, 21, 45, 32	7A: 6, 32, 46, 49		8C: 52, 9, 53, 40		6C: 21, 32, 45, 51	8C: 30, 5, 21, 45
6C: 6, 32, 45, 55	7A: 46, 50, 9, 3	8C: 21, 5, 39, 51				
						7A: 6, 32, 48, 51
	8C: 21, 52, 49, 40			7A: 23, 6, 53, 40		

SUBSTITUTION FOR TEACHER Ranganathan R

10A: 19, 3, 50, 52			9A: 4, 27, 42, 20				
			10A: 3, 43, 19, 53	9A: 42, 51, 50, 38			
	9A: 4, 27, 20, 49						10A: 43, 53, 6, 7
10A: 43, 24, 52, 47		9A: 20, 50, 52, 51					
				10A: 19, 3, 43, 53		9A: 4, 42, 27, 51	
		9A: 42, 38, 52, 47			10A: 24, 50, 54, 51		

SUBSTITUTION FOR TEACHER Geetha Ganesh

12B: 50, 26, 47, 3			12C: 36, 3, 4, 7			12C: 33, 36, 2, 5	
	12C: 33, 50, 2, 7			12B: 26, 50, 5, 6	12B: 26, 7, 8, 9		
	12B: 2, 50, 29, 26	12C: 33, 2, 3, 7	12C: 33, 35, 4, 6				
12C: 33, 36, 47, 6							12B: 2, 26, 4, 5
		12B: 2, 6, 7, 8			12C: 33, 50, 47, 36		
	12C: 33, 50, 7, 8			12B: 29, 26, 6, 7			

SUBSTITUTION FOR TEACHER Hemalatha P

11C: 33, 37, 50, 47		11B: 47, 3, 4, 5	12A: 28, 3, 4, 7	11A: 1, 6, 7, 8		11A: 2, 29, 5, 7	12A: 2, 38, 25, 6
11C: 2, 33, 37, 1	11A: 38, 2, 29, 50		11B: 26, 2, 3, 7	12A: 38, 50, 28, 5	11C: 7, 8, 9, 10		
	11A: 2, 29, 50, 4		12A: 38, 28, 4, 6	11B: 47, 28, 24, 4	11C: 2, 33, 37, 5		
11C: 33, 47, 36, 37	11C: 33, 50, 37, 3	12A: 28, 50, 1, 5			11A: 25, 3, 6, 7		11B: 26, 28, 2, 4
	11B: 47, 28, 7, 8	12A: 2, 28, 6, 7		11C: 33, 37, 50, 3	11A: 50, 47, 7, 9	11B: 26, 28, 4, 6	
12A: 38, 50, 47, 5		11C: 2, 33, 37, 47	11A: 38, 29, 25, 47	11B: 28, 26, 6, 7			

SUBSTITUTION FOR TEACHER Uma Sriram

	10C: 8, 53, 51, 52		10A: 3, 51, 49, 53		10B: 43, 50, 54, 51	10A: 12, 54, 49, 52	10D: 54, 38, 51, 8
	10B: 49, 16, 50, 38	10C: 42, 27, 52, 8		10D: 42, 38, 50, 51			10A: 43, 3, 16, 54
10A: 3, 52, 16, 38				10B: 24, 43, 16, 47		10C: 27, 53, 50, 51	10D: 53, 6, 7, 9
10C: 42, 47, 54, 51	10A: 3, 16, 12, 50		10B: 3, 54, 50, 51				10D: 27, 38, 54, 2
		10C: 42, 51, 12, 52			10A: 24, 54, 51, 16	10D: 42, 27, 54, 52	10B: 51, 8, 16, 54
	10C: 49, 8, 54, 50	10B: 43, 47, 52, 38	10A: 53, 47, 16, 12		10D: 27, 1, 47, 50	10B: 24, 43, 50, 51	

SUBSTITUTION FOR TEACHER Revathi Sivaram

	9A: 8, 47, 12, 51	9B: 3, 52, 47, 38		9C: 53, 52, 12, 39		9A: 42, 52, 49, 54	9D: 39, 40, 51, 54
9C: 4, 54, 39, 40	9D: 51, 8, 12, 53		9A: 53, 2, 3, 7	9C: 43, 39, 40, 53			9B: 3, 16, 53, 50
9C: 12, 39, 8, 52		9B: 3, 42, 53, 12	9D: 39, 40, 54, 52		9A: 27, 42, 16, 51		
9D: 43, 49, 47, 54	9A: 42, 4, 50, 16		9C: 52, 53, 50, 47	9B: 29, 16, 51, 8		9B: 8, 12, 52, 51	
		9C: 43, 12, 8, 53	9A: 53, 54, 16, 38	9B: 3, 52, 53, 50		9D: 27, 51, 39, 54	
9D: 27, 51, 50, 39	9A: 16, 53, 12, 49	9D: 43, 47, 52, 2	9C: 25, 4, 8, 53	9B: 29, 8, 12, 51			

SUBSTITUTION FOR TEACHER Jamuna N

	8D: 31, 46, 54, 39	8C: 45, 5, 15, 52		6C: 45, 6, 15, 55	7D: 44, 53, 39, 40	8A: 31, 53, 13, 52	
8A: 31, 4, 13, 41		7D: 40, 52, 48, 54		6C: 6, 45, 51, 55	8D: 46, 31, 9, 52	8C: 5, 48, 53, 54	
	7D: 31, 50, 52, 39	6C: 53, 15, 55, 51	8D: 31, 4, 48, 54	8C: 30, 53, 52, 15			8A: 31, 46, 48, 9
8C: 45, 40, 39, 51	8D: 31, 4, 46, 9		6C: 32, 54, 50, 53	7D: 5, 31, 54, 39			8A: 46, 4, 41, 40
	6C: 45, 32, 48, 55	8C: 54, 40, 39, 15		8A: 31, 50, 52, 41	7D: 31, 14, 54, 52		8D: 46, 4, 39, 41
8D: 46, 48, 50, 51		8A: 46, 41, 52, 9	8C: 5, 30, 9, 50		6C: 32, 45, 6, 54	7D: 31, 54, 48, 51	

SUBSTITUTION FOR TEACHER Indu Satish

	6B: 46, 51, 53, 52		7C: 51, 53, 49, 54		6A: 44, 32, 54, 39	7B: 30, 5, 13, 52	
6B: 6, 30, 48, 39	7B: 45, 48, 51, 54			6A: 5, 44, 39, 41		7C: 5, 44, 53, 14	6B: 53, 13, 50, 39
	7B: 30, 10, 49, 52		6A: 54, 48, 9, 52	6B: 30, 46, 11, 53		7C: 32, 5, 14, 53	
6A: 44, 32, 49, 39				6B: 51, 39, 41, 11	7B: 45, 52, 54, 10	7C: 44, 41, 48, 52	
	6A: 32, 49, 9, 48		6B: 46, 6, 53, 55	7B: 5, 45, 49, 53	7C: 53, 10, 40, 51		
7C: 5, 50, 14, 53			6A: 44, 5, 53, 48		7B: 30, 45, 51, 41	6B: 51, 13, 11, 55	

SUBSTITUTION FOR TEACHER Rajalekshmy Balaji

7A: 6, 32, 41, 15				6D: 30, 6, 45, 52	8B: 44, 39, 40, 50		
8B: 30, 4, 54, 48				7A: 46, 6, 53, 15	6D: 10, 39, 40, 55		8B: 53, 14, 54, 50
	6D: 6, 30, 55, 50					7A: 32, 53, 51, 9	8B: 44, 53, 14, 39
	8B: 44, 30, 4, 39			7A: 32, 54, 48, 15	6D: 45, 6, 54, 52	6D: 6, 30, 53, 55	
		6D: 6, 39, 40, 53			7A: 9, 51, 53, 54	8B: 44, 4, 30, 52	
	7A: 32, 46, 49, 40		6D: 30, 39, 40, 52		8B: 30, 48, 10, 54		

SUBSTITUTION FOR TEACHER Sreelatha Harish

11B: 28, 26, 47, 50	10A: 43, 8, 54, 52	10B: 3, 19, 51, 16	12B: 3, 4, 7, 10	10B: 19, 8, 52, 53	10A: 43, 51, 38, 47	11B: 2, 5, 7, 8	11B: 26, 2, 6, 7
11B: 1, 26, 2, 4	11B: 50, 2, 7, 8	10A: 50, 16, 52, 8	12B: 17, 2, 26, 29	11B: 28, 26, 50, 5	10A: 19, 16, 12, 52	12B: 17, 3, 5, 6	10B: 43, 3, 53, 16
12B: 17, 29, 26, 3	11B: 26, 50, 2, 4		10B: 19, 52, 16, 38		10A: 19, 43, 54, 16	12B: 17, 50, 26, 5	12B: 17, 26, 6, 7
	12B: 17, 50, 3, 4	11B: 1, 26, 50, 28	10A: 3, 47, 16, 53	11B: 18, 1, 28, 2	12B: 17, 26, 3, 6	12B: 2, 17, 26, 4	10B: 38, 54, 16, 2
	10B: 19, 43, 47, 49	10A: 43, 16, 49, 53	12B: 17, 2, 47, 1	11B: 50, 26, 28, 3		10B: 52, 54, 38, 51	12B: 17, 26, 4, 7
	11B: 18, 28, 26, 50	12B: 2, 17, 47, 26	10B: 12, 47, 16, 8	10A: 19, 43, 8, 51			

SUBSTITUTION FOR TEACHER Padma G

11A: 50, 47, 3, 6	12A: 18, 28, 47, 5	9C: 4, 39, 47, 52				9C: 49, 8, 53, 54	
11A: 2, 1, 4, 5	12A: 2, 38, 50, 7	12A: 18, 50, 4, 7	9C: 43, 53, 2, 3	11A: 38, 50, 5, 6	11A: 38, 7, 8, 9	12A: 18, 3, 5, 6	12A: 18, 50, 3, 7
12A: 38, 18, 3, 5	12A: 2, 50, 4, 6		11A: 38, 4, 6, 7	12A: 47, 28, 4, 7	9C: 43, 54, 51, 39	11A: 18, 50, 5, 7	
9C: 43, 54, 47, 52		11A: 50, 1, 5, 7	11A: 18, 50, 47, 3	12A: 18, 2, 28, 1		11A: 2, 18, 4, 6	
			11A: 2, 18, 38, 47	9C: 43, 51, 52, 49		12A: 38, 28, 4, 6	9C: 4, 20, 12, 51
11A: 38, 47, 50, 5		12A: 2, 38, 47, 28		9C: 43, 51, 53, 8			

SUBSTITUTION FOR TEACHER Vasanthy

	11B: 18, 47, 28, 5	12B: 17, 47, 3, 4			12B: 17, 50, 47, 7	12B: 2, 29, 5, 7	
	12B: 2, 50, 29, 7					11B: 18, 3, 5, 6	
		11B: 18, 24, 28, 2		12B: 17, 24, 47, 4	11B: 2, 5, 7, 9		
11B: 47, 24, 28, 6	11B: 50, 28, 3, 4		11B: 18, 28, 50, 47	12B: 2, 17, 29, 1			
			11B: 18, 1, 47, 2		12B: 24, 47, 50, 7		
12B: 17, 24, 47, 50			11B: 47, 50, 4, 5				

SUBSTITUTION FOR TEACHER Flora Bhaskar

10D: 42, 19, 53, 47					10C: 42, 51, 52, 38	9D: 49, 8, 53, 12	9A: 38, 54, 12, 16
	10C: 49, 38, 50, 53		10D: 19, 53, 2, 3	9D: 43, 51, 50, 53	9A: 42, 20, 12, 16		
10C: 3, 52, 12, 8		9A: 42, 8, 16, 53	10D: 19, 52, 53, 54	10D: 42, 52, 53, 47			9D: 43, 20, 53, 39
9A: 42, 49, 52, 47	10C: 42, 3, 50, 8	9D: 43, 20, 52, 50		10C: 19, 8, 12, 51		10D: 19, 8, 52, 12	
	10D: 42, 19, 8, 49	9D: 43, 12, 54, 49	10C: 19, 3, 53, 47		9D: 20, 53, 40, 54		9A: 4, 20, 12, 16
	9D: 53, 8, 52, 12	10C: 42, 47, 52, 38	9A: 4, 49, 47, 12	9A: 12, 16, 51, 53		10D: 42, 1, 51, 54	

SUBSTITUTION FOR TEACHER Vijaya Devanathan

		12A: 38, 47, 3, 4		11B: 1, 6, 7, 8	12A: 18, 38, 25, 50	11B: 2, 5, 7, 8	11B: 26, 2, 6, 7
11B: 1, 26, 2, 4	11B: 50, 2, 7, 8	11B: 18, 50, 26, 4	12A: 2, 3, 7, 9			12A: 18, 3, 5, 6	12A: 18, 50, 3, 7
12A: 18, 38, 3, 5	12A: 2, 50, 4, 6				12A: 2, 5, 7, 9	11B: 18, 26, 50, 5	
					11B: 26, 3, 6, 7	12A: 2, 18, 4, 6	
			12A: 18, 38, 2, 47		11B: 47, 50, 24, 7		11B: 18, 26, 4, 7
11B: 24, 50, 47, 5			12A: 38, 25, 50, 47				

SUBSTITUTION FOR TEACHER Kalavathy T

9B: 42, 3, 20, 51					11A: 18, 38, 50, 25		12B: 17, 2, 26, 6
9B: 54, 16, 1, 2		12B: 17, 50, 26, 4		11A: 38, 50, 5, 6	11A: 38, 7, 8, 9	11A: 18, 3, 5, 6	11A: 18, 50, 3, 7
		12B: 2, 24, 3, 7	9B: 8, 53, 16, 12	11A: 47, 4, 7, 10	12B: 17, 2, 5, 7	12B: 17, 50, 26, 5	12B: 17, 26, 6, 7
12B: 47, 24, 6, 7	9B: 3, 42, 16, 8	11A: 50, 1, 5, 7	11A: 18, 47, 50, 3		12B: 17, 26, 3, 6	12B: 2, 17, 26, 4	11A: 38, 2, 25, 4
		9B: 42, 52, 12, 54	9B: 3, 53, 47, 16			12B: 17, 26, 4, 6	11A: 18, 38, 4, 7
9B: 42, 38, 51, 53	12B: 50, 26, 7, 8	11A: 2, 38, 47, 6					

SUBSTITUTION FOR TEACHER Padma Balaji

8B: 53, 50, 52, 14	7B: 5, 45, 51, 53	6D: 23, 45, 6, 51			8C: 45, 51, 52, 53		6B: 22, 6, 11, 39
	8B: 23, 44, 49, 54	6D: 23, 50, 52, 54	7B: 22, 48, 41, 53		6B: 46, 22, 55, 11	7B: 5, 40, 53, 54	8C: 45, 21, 54, 15
8C: 45, 5, 21, 52		7B: 22, 51, 41, 53	6B: 6, 52, 11, 54		8B: 23, 51, 48, 54		6D: 6, 45, 39, 13
6D: 45, 6, 23, 39		6B: 46, 22, 11, 50	8B: 23, 44, 50, 53	7B: 45, 5, 13, 40	8C: 45, 21, 52, 54		
	7B: 45, 40, 41, 49	6B: 22, 6, 55, 39	8B: 23, 44, 40, 48	8C: 45, 5, 53, 49			6D: 23, 55, 10, 51
6D: 23, 55, 13, 51	6B: 46, 22, 53, 50	7B: 22, 13, 41, 52		8B: 23, 44, 39, 40		8C: 45, 48, 50, 54	

SUBSTITUTION FOR TEACHER Ramya R

7D: 21, 52, 50, 53			8A: 4, 46, 21, 13	8D: 46, 9, 52, 48	8D: 46, 52, 54, 14		
		8A: 46, 4, 52, 50		7D: 44, 5, 51, 10		8D: 14, 48, 54, 53	
8D: 21, 52, 39, 3		7D: 44, 53, 51, 14		8A: 4, 46, 41, 52		7D: 21, 5, 14, 53	
8A: 46, 52, 54, 41						8D: 4, 21, 52, 41	7D: 5, 39, 40, 54
			8A: 46, 21, 9, 54			8D: 4, 21, 39, 51	7D: 44, 10, 54, 14
7D: 5, 53, 39, 51		8D: 46, 52, 14, 9	8A: 4, 53, 41, 40				

SUBSTITUTION FOR TEACHER Padma Malini

	6C: 45, 40, 41, 52	7C: 44, 5, 22, 52	7A: 46, 23, 52, 40	6A: 44, 22, 52, 48			6A: 22, 49, 14, 51
6C: 45, 6, 15, 55	6A: 44, 50, 54, 53			7C: 44, 5, 50, 10		7A: 6, 23, 48, 15	
		7C: 22, 44, 41, 14	6C: 6, 48, 40, 41	7A: 23, 46, 53, 15			6A: 44, 22, 48, 14
	6A: 22, 44, 14, 39	6C: 21, 52, 51, 55			7C: 54, 10, 52, 41	7A: 6, 48, 9, 41	
			7C: 5, 44, 54, 40	6A: 5, 53, 50, 49	6C: 45, 40, 52, 51		7A: 46, 23, 40, 41
6C: 48, 51, 52, 55			7C: 5, 44, 49, 52	6A: 44, 22, 53, 39		7A: 23, 50, 51, 48	

SUBSTITUTION FOR TEACHER Usha Nandhini

	12C: 17, 47, 5, 7		11C: 37, 36, 3, 4				
		11C: 18, 50, 37, 36	11C: 2, 36, 37, 3	12C: 36, 50, 5, 6	12C: 35, 7, 8, 9		
12C: 17, 3, 5, 6							11C: 18, 37, 6, 7
		12C: 17, 1, 50, 36	11C: 18, 37, 50, 47	12C: 17, 1, 2, 5			11C: 2, 37, 36, 4
	12C: 17, 36, 47, 7		12C: 1, 17, 47, 2				11C: 18, 37, 4, 7
			12C: 17, 36, 47, 50	11C: 37, 6, 7, 8			

SUBSTITUTION FOR TEACHER Malabika Ghosh

		11D: 47, 3, 4, 5	11D: 37, 3, 4, 7	12D: 1, 36, 6, 7		11D: 37, 2, 5, 7	12D: 36, 35, 35, 2
		12D: 50, 36, 4, 7	11D: 37, 2, 3, 7				
	12D: 36, 50, 2, 4	12D: 2, 3, 7, 8				11D: 50, 37, 5, 7	
12D: 36, 47, 6, 7					12D: 3, 6, 7, 8	11D: 2, 4, 6, 7	
				12D: 50, 3, 5, 7			11D: 35, 37, 35, 4
		11D: 37, 47, 2, 6	12D: 47, 50, 36, 4				

SUBSTITUTION FOR TEACHER Sowmya Swamy

12C: 33, 47, 50, 3	11D: 34, 37, 47, 5	12D: 47, 36, 3, 4	12D: 36, 3, 4, 7	11D: 1, 6, 7, 8	12D: 34, 47, 50, 7	12D: 36, 2, 5, 7	
12C: 17, 1, 33, 2	12D: 34, 50, 2, 7	11D: 37, 50, 4, 7	12C: 17, 36, 2, 3	12D: 34, 36, 50, 5		11D: 34, 37, 3, 5	12D: 34, 50, 3, 7
11D: 34, 37, 3, 5	12C: 33, 50, 36, 2	11D: 37, 2, 3, 7		12D: 34, 36, 47, 4	11D: 34, 37, 2, 5	12D: 36, 50, 5, 7	11D: 34, 37, 6, 7
11D: 37, 47, 6, 7	11D: 34, 50, 37, 3	12D: 1, 34, 50, 36	11D: 34, 37, 50, 47	12D: 34, 1, 2, 5	12C: 17, 33, 3, 6	12C: 17, 33, 36, 2	11D: 34, 37, 2, 4
	12D: 34, 36, 47, 7	12C: 33, 2, 6, 7	11D: 34, 1, 47, 2	11D: 50, 37, 3, 5	12D: 34, 36, 50, 47		
12D: 34, 36, 47, 50	12D: 34, 50, 7, 8	12C: 17, 33, 47, 2	11D: 47, 50, 4, 5	11D: 34, 37, 6, 7			

SUBSTITUTION FOR TEACHER Anbazhagan K

12D: 34, 50, 47, 3	11C: 18, 47, 37, 5				12C: 33, 17, 50, 47		
12D: 1, 34, 2, 4	11C: 2, 33, 37, 50				12D: 34, 35, 35, 7	12C: 17, 33, 3, 5	12C: 17, 33, 50, 3
12D: 34, 3, 5, 6		11C: 33, 18, 2, 37	11C: 33, 4, 6, 7		12C: 33, 17, 2, 5		12C: 17, 6, 7, 9
	12C: 33, 17, 50, 3		12D: 34, 47, 50, 3	11C: 18, 2, 1, 5	11C: 33, 37, 3, 6		
		12D: 34, 2, 6, 7	12D: 1, 34, 47, 2	12C: 17, 33, 50, 3		11C: 33, 4, 6, 7	12D: 35, 35, 4, 7
	11C: 33, 18, 37, 50	12D: 47, 2, 6, 7		12C: 6, 7, 8, 10			

SUBSTITUTION FOR TEACHER Bhuvaneswari R

		11C: 33, 47, 36, 3		11C: 33, 36, 1, 6	11D: 34, 50, 47, 7		11D: 35, 35, 2, 6
					11D: 34, 35, 35, 7		11C: 33, 18, 50, 3
	11C: 2, 33, 50, 36		11D: 34, 35, 35, 4				
		11D: 34, 1, 50, 5		11D: 1, 34, 2, 5		11C: 18, 33, 2, 36	
	11C: 47, 36, 7, 8	11C: 2, 33, 6, 7	11C: 2, 18, 47, 1			11D: 34, 35, 35, 4	
11D: 34, 47, 50, 5			11C: 47, 50, 36, 4				

SUBSTITUTION FOR TEACHER Sekar K

10C: 3, 42, 19, 47	11A: 18, 47, 5, 7		10B: 3, 49, 52, 54	12A: 1, 6, 7, 8		12A: 2, 5, 7, 8	
12A: 2, 1, 4, 5		9B: 20, 42, 8, 16	11A: 2, 29, 3, 7			9A: 27, 20, 8, 54	10D: 27, 50, 53, 54
	10A: 19, 54, 52, 49	11A: 2, 18, 25, 3		9A: 42, 4, 20, 53	11A: 2, 5, 7, 9	12A: 18, 50, 5, 7	11A: 18, 25, 6, 7
12A: 47, 28, 6, 7		10A: 43, 19, 49, 51	12A: 18, 50, 47, 28	11A: 2, 18, 29, 1	9B: 20, 3, 54, 12	10B: 19, 8, 16, 53	
	11A: 47, 7, 8, 9	10D: 42, 8, 12, 51		12A: 50, 28, 3, 5	12A: 50, 47, 7, 9		
	11A: 18, 50, 25, 7			12A: 28, 6, 7, 8	10C: 27, 54, 50, 51		

SUBSTITUTION FOR TEACHER Sangeetha V

9D: 3, 20, 52, 51						8C: 5, 30, 49, 54	
	8C: 45, 21, 49, 50	8D: 46, 4, 48, 50	8B: 44, 23, 14, 48			6A: 44, 5, 54, 48	
		6D: 23, 53, 55, 13		8B: 30, 44, 23, 4		6B: 30, 53, 11, 51	
			7D: 31, 44, 50, 54		9C: 43, 25, 20, 12	9D: 51, 53, 12, 8	
			6A: 44, 5, 54, 53		6B: 30, 50, 11, 51		
	8D: 46, 21, 31, 53	9C: 43, 52, 47, 2			7D: 31, 10, 54, 50	6D: 45, 23, 13, 50	

SUBSTITUTION FOR TEACHER Meenakshi Venkatraman

9D: 3, 20, 50, 51		6C: 6, 45, 15, 51		7B: 22, 30, 45, 52		8C: 30, 5, 48, 15	
	8C: 45, 21, 49, 54		8B: 44, 23, 48, 14				
7A: 23, 32, 6, 52	7C: 32, 49, 10, 52	6D: 23, 13, 53, 55		8B: 23, 4, 30, 44	7B: 22, 5, 45, 13		6C: 6, 45, 53, 48
	8A: 4, 31, 46, 9	7C: 22, 5, 52, 51	7D: 31, 44, 52, 51		9C: 43, 20, 25, 52	9D: 51, 8, 52, 53	
						8A: 21, 4, 52, 54	
7A: 46, 23, 52, 50		9C: 43, 52, 47, 2			7D: 31, 54, 10, 51	6D: 23, 45, 55, 50	

SUBSTITUTION FOR TEACHER Sivagami M

		6C: 6, 45, 51, 55		7B: 22, 30, 45, 48			
		8D: 46, 4, 54, 52				6A: 44, 5, 53, 14	
7A: 32, 6, 23, 52	7C: 32, 52, 49, 54				7B: 45, 5, 22, 13	6B: 30, 55, 50, 53	6C: 6, 45, 48, 15
	8A: 4, 31, 46, 9	7C: 5, 22, 50, 14					
			6A: 44, 5, 48, 53		6B: 30, 53, 11, 52	8A: 21, 4, 48, 52	
7A: 46, 23, 52, 53	8D: 21, 46, 31, 52						

SUBSTITUTION FOR TEACHER Samundeeswari K

	10D: 54, 53, 8, 51	9A: 4, 51, 38, 47		9B: 12, 52, 53, 16			10C: 38, 51, 54, 49
9A: 4, 27, 54, 16	10D: 12, 50, 54, 8		10C: 3, 19, 53, 2			9B: 3, 20, 12, 8	10C: 3, 27, 50, 54
9B: 29, 3, 52, 8	10D: 27, 19, 49, 50		9A: 4, 54, 52, 38			9B: 20, 53, 8, 12	10C: 53, 6, 7, 9
		10D: 1, 19, 52, 49		10D: 19, 1, 8, 51	10C: 27, 3, 19, 52	9A: 4, 12, 16, 52	9B: 20, 38, 16, 54
			10D: 19, 1, 54, 38	10C: 19, 3, 49, 52	9A: 20, 51, 16, 53		9B: 20, 12, 51, 54
	10D: 49, 52, 8, 50		9B: 29, 16, 53, 8	10C: 19, 51, 53, 12	9A: 20, 27, 47, 51		

SUBSTITUTION FOR TEACHER Uma Rameswaran

9C: 20, 47, 52, 53		9D: 3, 47, 52, 39	9C: 4, 20, 52, 49	9D: 27, 53, 52, 51		10B: 54, 49, 16, 12	10A: 51, 8, 12, 49
10B: 19, 16, 54, 1	10A: 8, 54, 50, 51	9D: 20, 27, 40, 8			9C: 20, 12, 52, 8	10B: 3, 19, 12, 53	
9D: 3, 12, 8, 39	10B: 19, 54, 52, 49	9C: 25, 8, 51, 53	10A: 19, 38, 16, 12			10A: 51, 16, 53, 12	
	10B: 3, 38, 8, 16		9D: 3, 27, 47, 51			9C: 4, 53, 12, 8	10A: 38, 54, 16, 2
			10A: 19, 3, 47, 54		10B: 24, 51, 54, 47	9C: 4, 51, 39, 52	9D: 20, 51, 54, 39
10A: 19, 24, 16, 50	10B: 12, 49, 53, 52		9D: 12, 52, 53, 39		9C: 25, 20, 51, 50		

SUBSTITUTION FOR TEACHER Rashmi Ramdas

6A: 22, 32, 14, 50	7D: 5, 31, 54, 39		8B: 23, 30, 4, 49				7C: 22, 49, 41, 40
6A: 5, 48, 39, 41		8B: 23, 4, 52, 48			7C: 22, 32, 41, 40		7D: 21, 31, 39, 40
	6A: 32, 9, 39, 54		7D: 31, 39, 40, 52		7C: 32, 22, 5, 48	8B: 30, 40, 14, 50	
		7D: 31, 21, 5, 14		7C: 5, 32, 48, 54	6A: 52, 14, 41, 54		8B: 30, 4, 23, 54
	7D: 31, 49, 39, 40	7C: 22, 32, 51, 52		8B: 23, 53, 39, 40	6A: 50, 51, 54, 14		
8B: 23, 52, 50, 14	7C: 32, 22, 49, 54	7D: 14, 10, 52, 2			7C: 32, 41, 48, 54	6A: 41, 50, 54, 51	

SUBSTITUTION FOR TEACHER Uma Maheswari

8C: 21, 9, 15, 52			6C: 21, 52, 40, 41			6D: 23, 30, 54, 52	7B: 22, 49, 51, 40
		7B: 22, 40, 54, 52	6C: 21, 32, 15, 55		8C: 9, 15, 52, 40	6D: 6, 23, 13, 40	
	6C: 32, 6, 50, 52	8C: 9, 51, 53, 15	6D: 23, 6, 48, 54	7B: 30, 41, 10, 13			
	6C: 50, 10, 55, 3	6D: 23, 51, 39, 50	8C: 50, 52, 53, 15			8C: 21, 30, 48, 9	7B: 5, 30, 22, 13
		6C: 6, 32, 40, 41	8C: 21, 5, 54, 9			6D: 6, 30, 48, 51	7B: 22, 51, 10, 40
8C: 5, 53, 51, 48	6C: 21, 32, 40, 55	6D: 23, 6, 52, 10	7B: 5, 30, 53, 52	6C: 21, 6, 51, 55			

SUBSTITUTION FOR TEACHER Nagalakshmi V

8D: 21, 9, 51, 52		6B: 6, 22, 51, 13				7A: 32, 23, 52, 15	8A: 21, 31, 54, 41
7A: 6, 54, 15, 40	8D: 21, 31, 48, 41					6B: 6, 13, 53, 11	8A: 31, 21, 53, 54
6B: 6, 22, 11, 39		8A: 9, 41, 13, 53	7A: 6, 23, 48, 40			8D: 21, 14, 51, 50	
			8A: 31, 53, 50, 54	8D: 31, 51, 54, 39	7A: 6, 15, 52, 41	6B: 30, 6, 41, 55	
		7A: 6, 32, 52, 51		8D: 31, 39, 41, 53	8A: 31, 52, 50, 9	6B: 30, 6, 22, 52	
			8D: 4, 53, 9, 48	6B: 6, 22, 13, 11	7A: 32, 6, 51, 9	8A: 4, 31, 48, 51	

SUBSTITUTION FOR TEACHER Rathinakumar

			9D: 27, 3, 20, 51	12C: 17, 33, 1, 36		11C: 33, 2, 36, 37	11A: 2, 38, 25, 6
12B: 2, 17, 26, 1	9C: 53, 8, 49, 54	9A: 20, 4, 27, 42	9B: 29, 3, 53, 2	10C: 42, 50, 51, 38	11B: 26, 28, 7, 8	10D: 19, 27, 8, 53	11D: 34, 50, 3, 7
11C: 18, 37, 3, 5	9C: 20, 4, 50, 39	9D: 3, 8, 12, 51	11B: 28, 26, 4, 6		9B: 42, 16, 51, 54	10B: 8, 16, 51, 12	12D: 34, 6, 7, 9
	12A: 38, 25, 50, 28	12B: 17, 26, 50, 1		10B: 19, 43, 8, 12	9A: 20, 27, 12, 52	12D: 36, 2, 4, 6	12A: 38, 2, 25, 28
		11D: 34, 2, 6, 7		11A: 50, 3, 5, 7		12C: 17, 33, 35, 4	10A: 8, 16, 51, 12
	10A: 53, 12, 50, 16			10D: 19, 53, 12, 51		10C: 42, 38, 54, 50	

SUBSTITUTION FOR TEACHER Esther Rani

6C: 32, 6, 21, 51	8C: 5, 45, 40, 39	6A: 44, 5, 22, 14			7B: 45, 52, 13, 51	7D: 31, 21, 54, 49
				8C: 5, 45, 30, 15	8B: 10, 39, 40, 52	7C: 32, 50, 54, 53
8A: 21, 52, 3, 5	6B: 30, 6, 46, 55	7A: 23, 51, 15, 9		6C: 55, 52, 15, 10		6D: 30, 45, 50, 51
	7B: 22, 30, 10, 50	8A: 21, 31, 46, 49	7A: 23, 32, 41, 52		8D: 21, 31, 54, 41	6A: 5, 32, 22, 39
		8B: 54, 53, 51, 39		6D: 45, 23, 52, 53	8D: 31, 51, 9, 14	
	7D: 31, 21, 50, 53	6B: 46, 22, 6, 52		7C: 44, 22, 53, 14		

SUBSTITUTION FOR TEACHER Vijaya Lakshmi

9A: 20, 42, 52, 50	8B: 23, 54, 51, 39	7A: 23, 6, 51, 15		7A: 6, 46, 9, 48	10D: 42, 53, 47, 50		
10A: 19, 54, 16, 1		7C: 32, 22, 52, 54	7D: 31, 44, 21, 14	10B: 43, 50, 38, 53	9B: 42, 20, 52, 12	9D: 20, 27, 3, 8	6A: 32, 9, 14, 39
7B: 45, 5, 22, 52		10C: 19, 3, 42, 51	8A: 4, 31, 53, 13	7D: 44, 53, 52, 10	8D: 46, 31, 21, 9	6A: 5, 32, 51, 14	8D: 31, 46, 39, 14
	9D: 3, 50, 8, 39		9B: 3, 42, 53, 47	9C: 43, 12, 39, 40	10A: 3, 43, 19, 54	8B: 4, 44, 30, 51	7C: 22, 32, 5, 41
			10B: 3, 19, 38, 16		8C: 30, 45, 9, 53	10C: 42, 27, 54, 52	10D: 12, 38, 51, 8
9C: 47, 52, 51, 50		8C: 15, 52, 9, 2		7B: 22, 13, 40, 41	8A: 31, 9, 48, 54	9A: 4, 42, 20, 54	

SUBSTITUTION FOR TEACHER Govindarajulu

						9B: 42, 29, 49, 53	
6D: 6, 30, 45, 55		9C: 4, 20, 8, 50	8C: 21, 48, 53, 15		10B: 19, 52, 16, 12	8B: 44, 23, 48, 40	8D: 31, 21, 53, 39
9A: 12, 52, 16, 8	9D: 20, 27, 52, 54		7C: 53, 52, 10, 54	10C: 42, 52, 47, 53			7B: 45, 22, 48, 53
	7D: 44, 31, 39, 10				10D: 19, 27, 12, 52		6C: 6, 32, 15, 54
	10A: 19, 43, 49, 8		7A: 23, 6, 46, 48				
	8A: 31, 46, 21, 9	6A: 22, 32, 41, 52	6B: 30, 53, 50, 39				

SUBSTITUTION FOR TEACHER Chandrasekar

							9C: 25, 54, 39, 40
7D: 5, 31, 54, 48	6B: 55, 41, 11, 50		7A: 32, 46, 23, 15	7B: 5, 45, 30, 10		10C: 3, 19, 27, 8	6D: 45, 39, 40, 54
		10B: 19, 24, 3, 8			9D: 27, 43, 54, 39	10D: 27, 8, 50, 12	9A: 20, 53, 6, 7
	7C: 44, 22, 10, 50			8B: 54, 51, 39, 40			8C: 5, 30, 54, 15
	8A: 31, 46, 49, 9		6C: 6, 21, 53, 15				6A: 44, 22, 14, 48
				8D: 21, 31, 39, 41	9B: 20, 29, 54, 47	10A: 43, 24, 54, 16	

SUBSTITUTION FOR TEACHER Premabai B

		8A: 31, 4, 13, 52					9B: 8, 38, 54, 16
7B: 5, 30, 45, 54		10D: 27, 42, 52, 50			9D: 20, 39, 40, 12		
7D: 44, 21, 5, 39	8D: 31, 46, 4, 54				8C: 5, 45, 21, 39		
6B: 6, 46, 54, 48	9C: 4, 25, 12, 50	10C: 19, 52, 49, 51		10A: 43, 19, 51, 54	10B: 19, 43, 3, 12		9A: 27, 4, 20, 38
	7C: 32, 48, 41, 40					6A: 22, 32, 44, 52	6C: 15, 51, 54, 10
		7A: 23, 32, 46, 6			6D: 6, 30, 45, 54	8B: 4, 23, 54, 51	

SUBSTITUTION FOR TEACHER Vidhya Bharathi

7C: 32, 22, 41, 53		8B: 23, 44, 4, 52		9A: 27, 52, 8, 53			
			9D: 3, 43, 53, 2	8A: 46, 41, 40, 50	7A: 32, 46, 40, 41		
10B: 3, 52, 8, 16		10A: 24, 19, 3, 16		9B: 20, 42, 52, 53		9C: 20, 12, 8, 50	8C: 45, 9, 53, 39
	10D: 42, 12, 8, 38	8D: 21, 31, 46, 51				7B: 30, 48, 41, 53	
	10C: 42, 19, 49, 8			6B: 52, 13, 51, 39			
6A: 5, 52, 39, 48		6C: 6, 32, 15, 55	7D: 44, 5, 49, 48	6D: 6, 23, 10, 13			

SUBSTITUTION FOR TEACHER Vrinda Unnikrishnan

					6D: 45, 51, 54, 13		
						6C: 6, 53, 54, 40	
					6D: 45, 23, 54, 13		6B: 6, 22, 46, 13
					6C: 45, 21, 6, 41		6B: 30, 46, 6, 22

SUBSTITUTION FOR TEACHER Aishwarya J

	9B: 16, 51, 12, 52	7D: 44, 5, 31, 52	7B: 30, 10, 48, 51	8C: 30, 45, 39, 51		8D: 31, 54, 52, 49	11C: 2, 33, 36, 6
11D: 1, 34, 37, 2			8A: 21, 46, 31, 9		6A: 22, 32, 9, 14	10A: 19, 3, 16, 53	
11B: 18, 26, 3, 5		12A: 2, 18, 28, 25	9C: 4, 54, 53, 39	12C: 33, 17, 47, 36	6B: 46, 22, 48, 13		7A: 6, 46, 53, 48
7C: 44, 32, 49, 51				9A: 16, 8, 54, 51	9D: 27, 20, 3, 43	10C: 19, 12, 52, 51	12D: 34, 36, 2, 4
	12B: 17, 47, 7, 8	11A: 2, 6, 7, 8	6D: 6, 23, 48, 55			6C: 32, 6, 21, 10	8B: 44, 4, 23, 14
		10D: 42, 52, 47, 38		10B: 43, 19, 12, 16			

9. Limitations of the Project

1. The algorithm used in the program is not very efficient. Since most of the program is based on trial and error, the complexity and time required for computing increases exponentially with larger data inputs. This is why we were unable to extend the program to all classes of the school and had to restrict it to 6th to 12th standards alone.
2. The program runs on the command line which is not comfortable for most users, unlike graphical user interfaces or websites.
3. The code is rigid in some ways. The user cannot change the subjects nor the frequency of the subjects for a particular class, which means that the code would only be useful for our school.
4. The code does not take into account the time allotted for each class. Each period of each class is assumed to last the same amount of time.
5. Since the program was specifically designed for the classes of our school, future modifications to functionality to incorporate more customisation will require re-writing significant portions of the code.
6. Multiple campus functionality is not available in the program. For example, our yoga teacher takes classes at the primary school on two days of the week and for the secondary school on the other days. The teacher cannot be expected to be available on both campuses on the same day.
7. Each teacher is allowed to teach only one subject. When we did require functionality for more than one subject, we had to create a new file to store the details which is not very efficient for larger scale applications.

10. Future Scope for Improvement

1. We could have made the program more user-friendly by making it available on a website of some form, or by using a graphical user interface with various libraries like tkinter, which would've been more easily accessible and visually appealing.
2. Our program showed an error at a certain point because some requirements weren’t met. To overcome this, we ran the code multiple times within the script until all requirements were met. We believe that this was a drawback since we were not able to fix the code due to the randomness of the error. In this regard, principles of recursion and artificial intelligence can be incorporated, so that the program can intelligently identify a single small correction to be made in the time tables already generated, instead of restarting the entire program.
3. More efficient algorithms, such as the genetic algorithm, can be incorporated to improve efficiency.
4. Functionality can be added to allow users to schedule various different subjects on their own without requiring editing the python script.

11. Bibliography

- Computer Science with Python (Book by Sumita Arora for class 11)
- Computer Science with Python (Book by Sumita Arora for class 12)
- 2021 Complete Python Bootcamp From Zero to Hero in Python by Jose Portilla - Udemy Course
- Stack Overflow to tackle minute problems during coding stackoverflow.com
- Saving python program's output on a text file https://stackoverflow.com/questions/25023233
- PEP8 Style Guide: https://pep8.org/
- Raise Custom errors / exceptions https://devtut.github.io/python/raise-custom-errors-exceptions.html#custom-exception
- Microsoft Visual Studio Code - Code editor: code.visualstudio.com
- Creately, to design the directory structure diagram: creately.com