NAME: DAYEETA DAS
NTU ID: N0830182

TITLE OF THE PROJECT:

CONTACT MANAGEMENT SYSTEM

SOFTWARE USED:

VISUAL STUDIO IDE

PROGRAMMING LANGUAGE USED:

C++

AIM OF THE PROJECT

The aim of this project is to create a Contact Management System that will enable the user to do various tasks using the records in a phonebook.

This program enables the user to select from a menu of 5 different tasks such as adding a contact, displaying all the contacts, searching for a contact, editing a contact and deleting a contact. It goes through all possible checks to avoid any sort of invalid input thus avoiding any form of errors. If the user enters an invalid number(that is a number which is not present in the menu) the program keeps asking for an input till it gets a valid response. Different tasks are performed based on the number put as an input by the user.

This program makes use of various C++ features such as **Class, arrays, pointers, functions, iteration and branching statements and extensive use of file handling.**

REQUIREMENTS OF THE PROGRAM

Here are a list of all the different elements(constants, variables, functions and so on) that are used within the program.

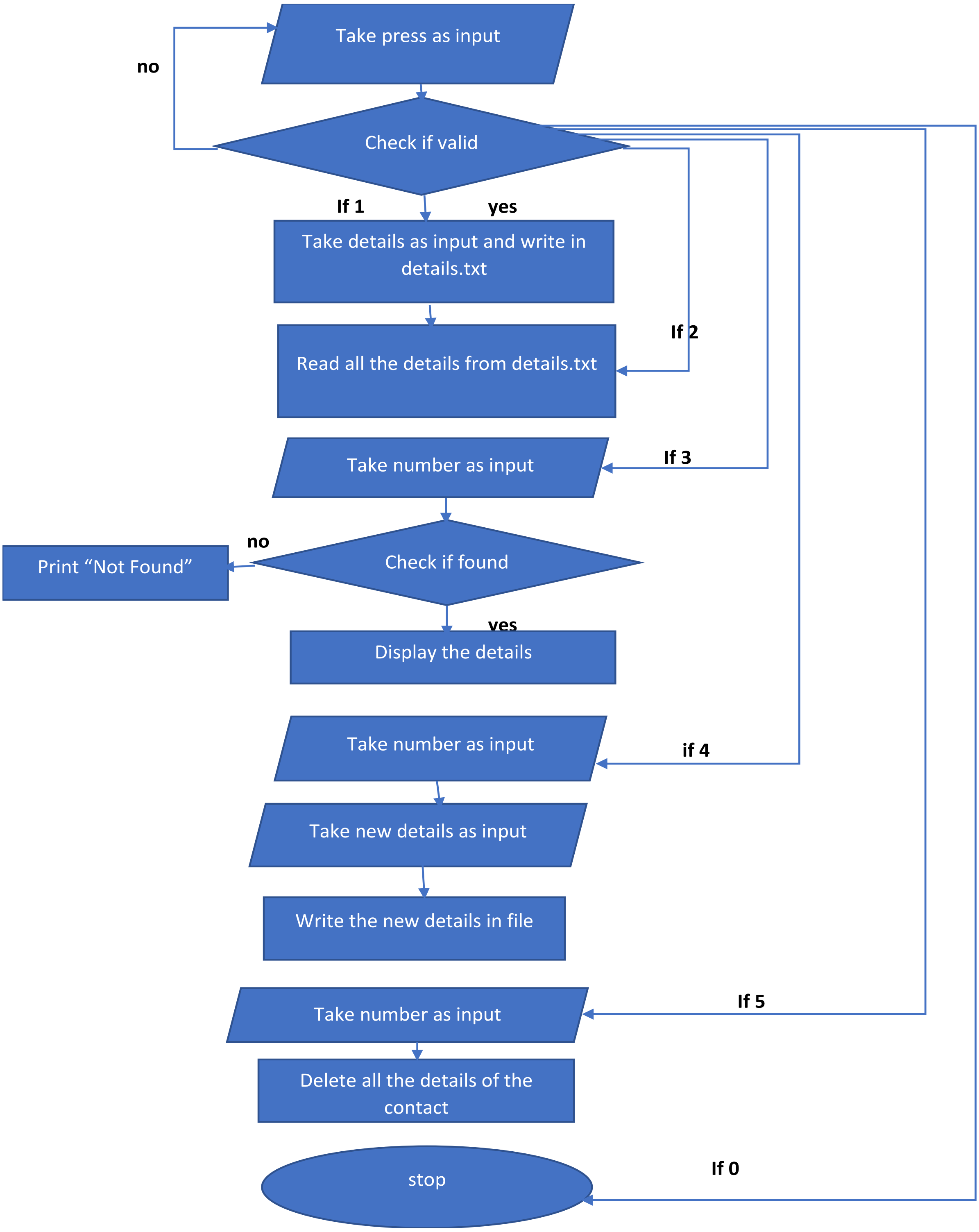
1. Header files used: <iostream>, <string.h>, <fstream>, <string>
2. Identifiers used(along with the data types):
    - Character type: press, name[20], email[30]
    - String type: phone, num
    - Boolean type: found
3. Class:
    - Name: Manage
    - Private members:
      phone, name[20]
      email[30]
    - Public members:
      void menu()
      void input_press()
      void check_press()
      void add_contact()
      void save_contact()
      void display()
      void display_contact()
      void search_contact(string)
      void edit_contact()
      void delete_contact()
4. Other data types used(with identifier names):
    fstream fp
    sizeof Manage
5. Statements used:
    Branching: if-else
    Iteration: while loop
    Input/ Output statements: cin , cout, cin.ignore()
    File Handling Statements: fp.open(), fp.close(), fp.read(), fp.write(), ios::in, ios::out, ios::app
    Break statement

FUNCTIONALITY OF THE PROGRAM

1. The necessary header files are included at the beginning of the program. Global variables are declared.
2. The class is defined with the private and public members.
3. The menu is displayed on the screen.
4. Input is taken from the user and its validity is checked.
5. If the user presses ‘1’ then the details: name, contact number and email are taken as input and written in the text file details.txt.
6. If the user presses ‘2’ then all the contact details is read from details.txt and displayed on the screen.
7. If the user presses ‘3’, the phone number to be searched is taken as input from the user and the contact details are displayed on the screen.
8. If the user presses ‘4’ the phone number is taken as input and the new details are taken as input for editing the existing contact.
9. If the user presses ‘5’ the phone number is taken as input and all the contact details of the particular number is deleted.
10. The main function is declared. The functions menu(), input_press() and check_press() are looped till the user presses ‘0’ which terminates the entire program.

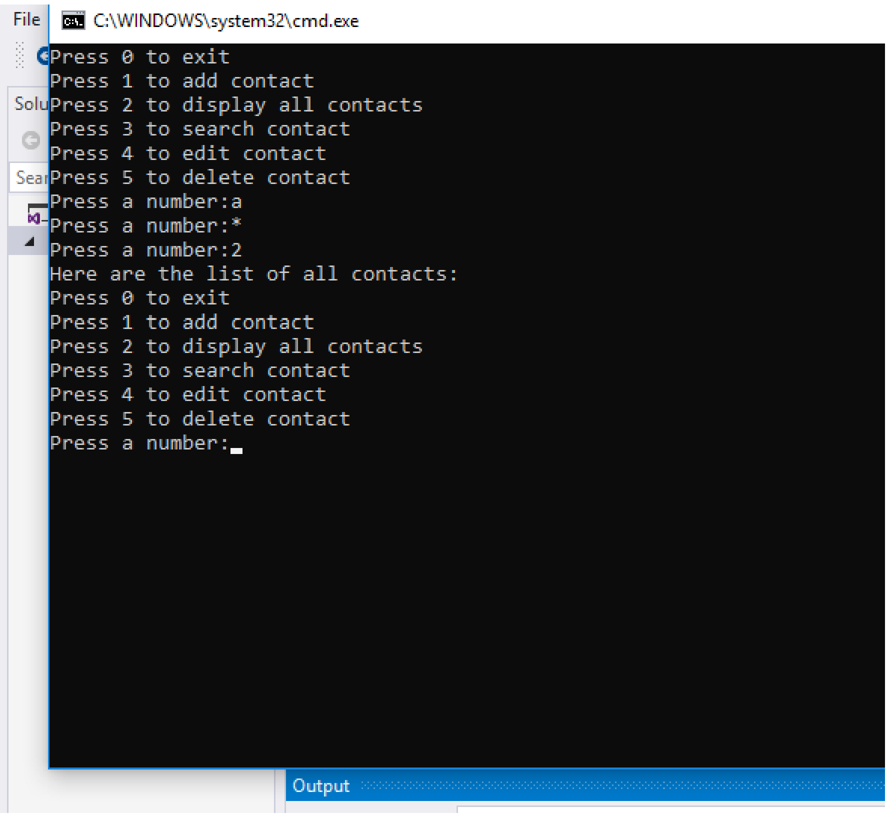
NAME: DAYEETA DAS
NTU ID: N0830182

FLOWCHART ILLUSTRATING THE WORKING OF THE PROGRAM



CHECKS UNDERTAKEN FOR MAKING THE PROGRAM EFFICIENT

- The void check_press() function. It checks if the number being taken as input from the user is within the range of 0 to 5 or not thus checking its validity. If the user enters an alphabet or a special character then the input is taken till it is valid.



```
Press 0 to exit
Press 1 to add contact
Press 2 to display all contacts
Press 3 to search contact
Press 4 to edit contact
Press 5 to delete contact
Press a number:a
Press a number:*
Press a number:2
Here are the list of all contacts:
Press 0 to exit
Press 1 to add contact
Press 2 to display all contacts
Press 3 to search contact
Press 4 to edit contact
Press 5 to delete contact
Press a number:_
```

CRITIQUE ON THE PROGRAM

- The contact details are taken as input in form of string and there are no checks performed to ensure that it is a valid contact number or a valid email address.
- Since the name is taken as input in form of a character array giving any space causes the program to crash.
- A check can be put to test if the contact to be edited or deleted exists or not.

Last but not the least I would like to conclude the report by stating that every project undertaken adds a new experience and each project can be improved even further if new ideas are added to it.