University of Liberal Arts Bangladesh

(Open Ended Lab Report-2)

Course Code	**:**	**CSE1302**
Course Title	**:**	**Data Structures Lab**
Section	**:**	**03**
Submitted to	**:**	**Shakib Mahmud Dipto**
Submitted by	**:**	**Kawser Ahmmed**
ID	**:**	**231014036**

Problem: You are brought on board to oversee the maintenance of a service management system. This system is responsible for storing vital information about customers, such as their name, age, gender, contact number and email address. However, during your review of the current setup, you noticed that all customer data is currently consolidated in a single repository.

1: Select and implement the most suitable data structure for the service management system. Justify your choice with a clear explanation.

2: The administration wants to maintain separate queues for male and female customers. To achieve this, they need two dedicated repositories exclusively for female and male customers. Now, your task is to create and develop this system to fulfill their requirement.

3: Develop a system for appointment management, where a customer can book an appointment for a specific day. For this system, you need to implement an appropriate data structure to manage the appointments. In this system, the customers will be served on a first-come, first-served basis.

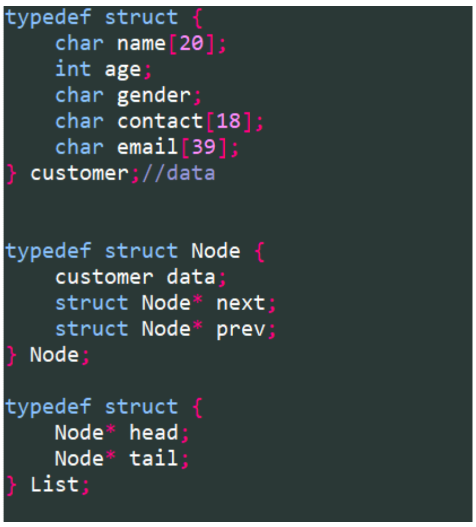
Datastructure:

To develop this service management system, I used two different doubly linked lists for two repositories, male and female. I think a doubly linked list is a suitable data structure for this type of service management system because we can easily delete any data from the list, print any data, and search any data. On the other hand, a doubly linked list is more beneficial than a singly linked list because it adds and deletes easily and efficiently.

For the appointment part, I have used a queue data structure, which is known as first in, first out, which is suitable for this type of serial maintenance data. In this programme, the queue is array-based, which is easy to handle, and we also use the front and rear to measure everything clearly.

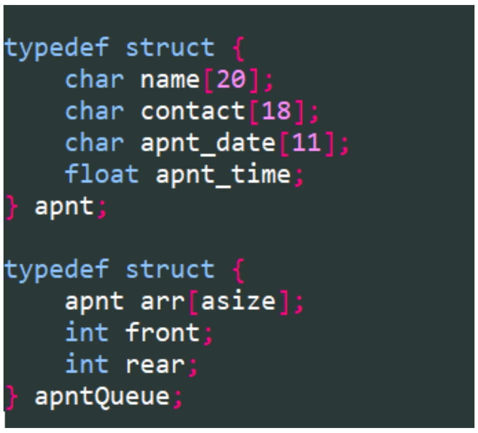
Structures

 I used all this structure to keep things organized.



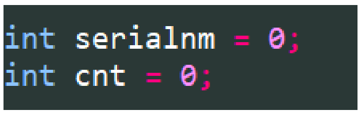
```
typedef struct {
    char name[20];
    int age;
    char gender;
    char contact[18];
    char email[39];
} customer;//data


typedef struct Node {
    customer data;
    struct Node* next;
    struct Node* prev;
} Node;

typedef struct {
    Node* head;
    Node* tail;
} List;
```

Here is the entire structure to create a proper linked list that I have used. First of all, struct customer is a type of data, then I created a struct node, which will be a node, and then the struct on list, which is used to create linked lists, I named it list so that I can easily use it (**NOTE**: typedef use to change structure's name so that I can easily write it and use it).



```
typedef struct {
    char name[20];
    char contact[18];
    char apnt_date[11];
    float apnt_time;
} apnt;

typedef struct {
    apnt arr[asize];
    int front;
    int rear;
} apntQueue;
```

This is the second structure I have created for my queue. The first apnt is for the data type, which I have used to create an array in the queue.

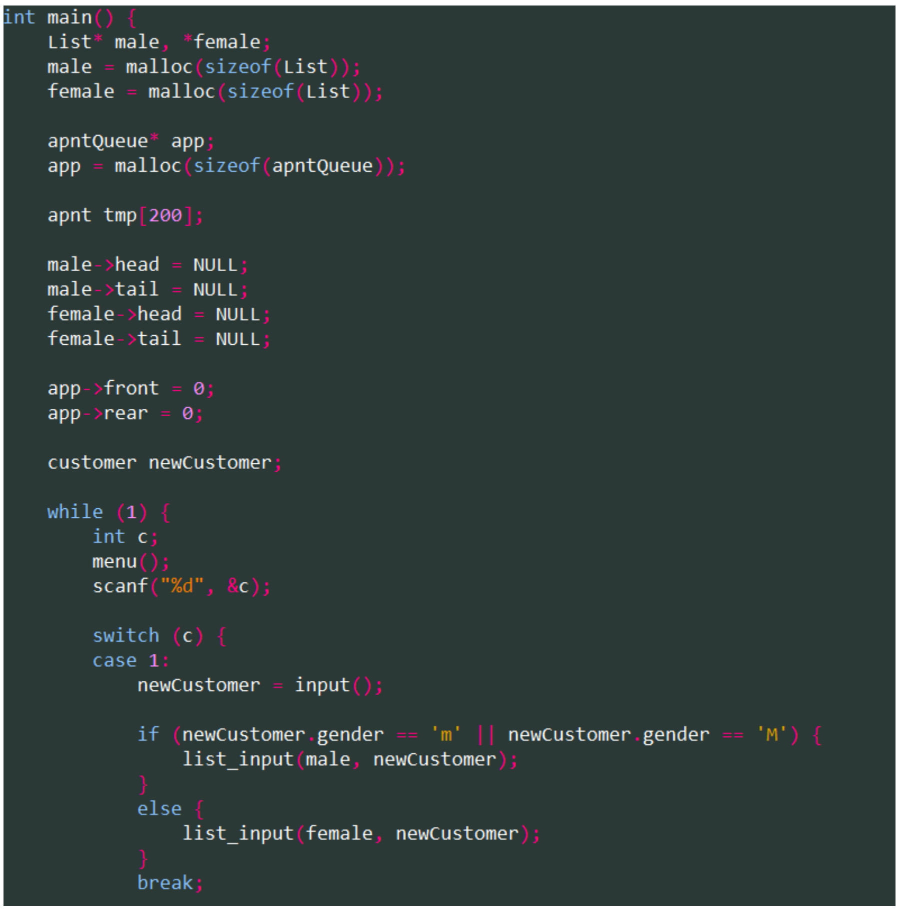


```c
int serialnm = 0;
int cnt = 0;
```

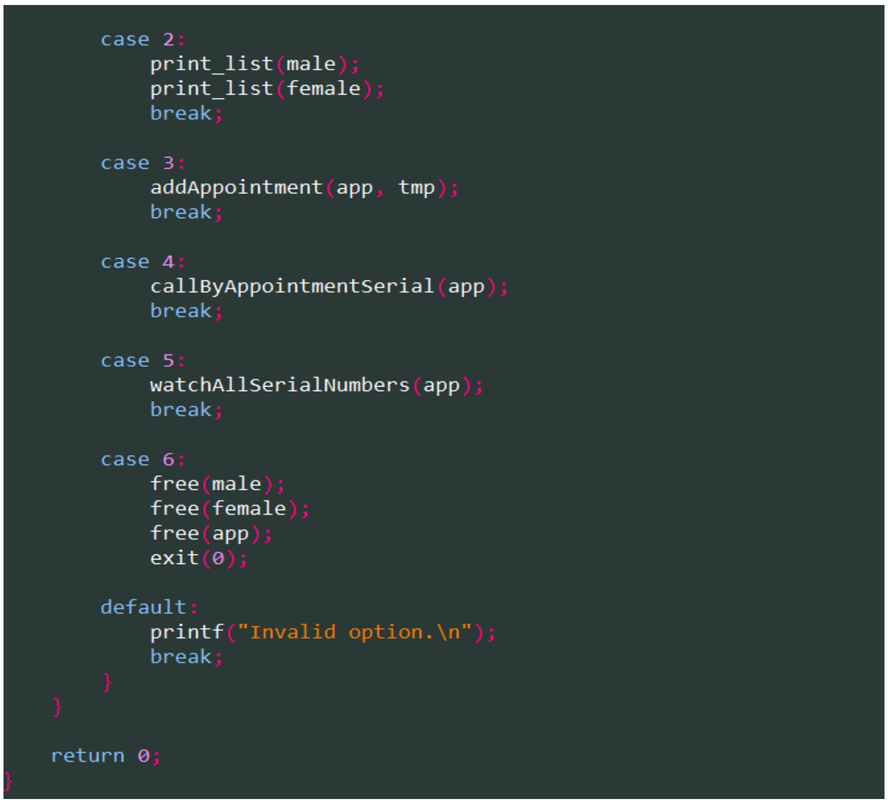
And this two are global variable for cout customer’s data nd appointment serial.

<u>Main function()</u>

In the main function, I have created a menu for this service management system so that the user can Easily understand the user manual and operate the system quickly.



```c
int main() {
    List* male, *female;
    male = malloc(sizeof(List));
    female = malloc(sizeof(List));

    apntQueue* app;
    app = malloc(sizeof(apntQueue));

    apnt tmp[200];

    male->head = NULL;
    male->tail = NULL;
    female->head = NULL;
    female->tail = NULL;

    app->front = 0;
    app->rear = 0;

    customer newCustomer;

    while (1) {
        int c;
        menu();
        scanf("%d", &c);

        switch (c) {
        case 1:
            newCustomer = input();

            if (newCustomer.gender == 'm' || newCustomer.gender == 'M') {
                list_input(male, newCustomer);
            }
            else {
                list_input(female, newCustomer);
            }
            break;
```


```
        case 2:
            print_list(male);
            print_list(female);
            break;

        case 3:
            addAppointment(app, tmp);
            break;

        case 4:
            callByAppointmentSerial(app);
            break;

        case 5:
            watchAllSerialNumbers(app);
            break;

        case 6:
            free(male);
            free(female);
            free(app);
            exit(0);

        default:
            printf("Invalid option.\n");
            break;
        }
    }

    return 0;
}
```

Fisr of all I declared two linked list (using previous structure) male and female and allocate memory for them and also declared q queue to store appointment data only queue alos a pointer so I also allocate memory for the queue and then the initialaization part I have initialize linked list male and female head to NULL and queue's front and rear =0

<u>All Function ()</u>

I have write down all the function I have used in my program:

1. **void menu();**

2. **customer input();**

3. **void list_input(List* list,customer newCustomer);**

4. **void addAppointment(apntQueue* app, apnt tmp[]);**

5. **void callByAppointmentSerial(apntQueue* app);**

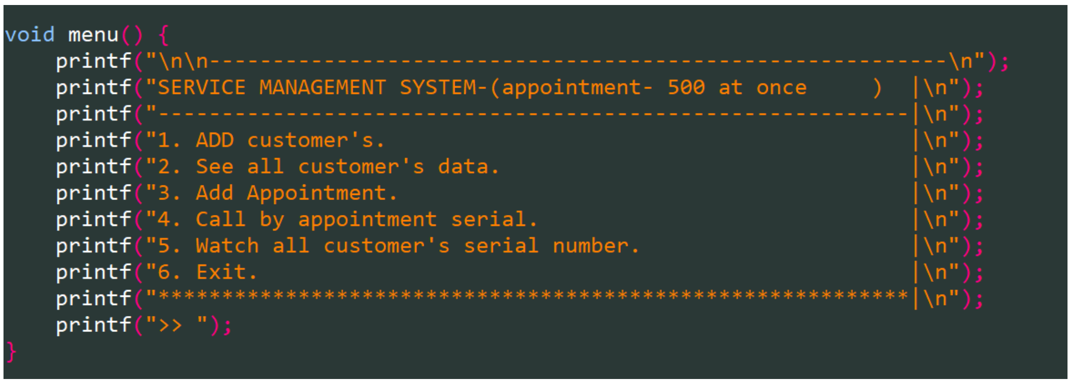
6. **void watchAllSerialNumbers(apntQueue* app);**

7. **void sortAppointmentQueue(apntQueue* app);**

8. **void print_lsit(List* list);**

<u>Menu function</u>

This menu function prints a user-friendly menu to the console, presenting various options for interacting with the service management system. These options include adding customers,

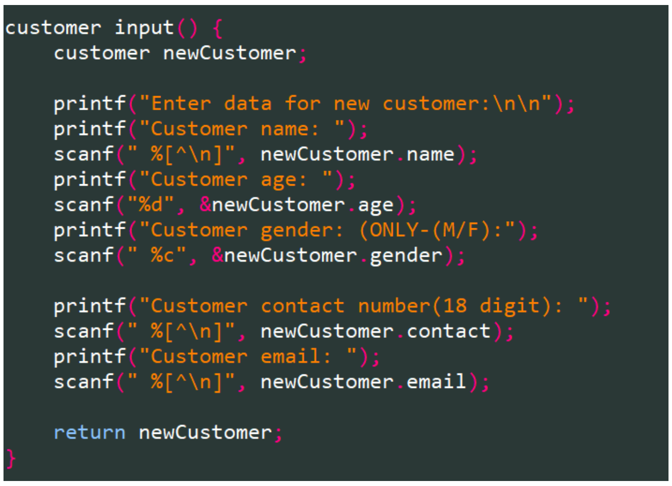


```c
void menu() {
    printf("\n\n----------------------------------------------------------\n");
    printf("SERVICE MANAGEMENT SYSTEM-(appointment- 500 at once     )  |\n");
    printf("-----------------------------------------------------------|\n");
    printf("1. ADD customer's.                                         |\n");
    printf("2. See all customer's data.                                |\n");
    printf("3. Add Appointment.                                        |\n");
    printf("4. Call by appointment serial.                             |\n");
    printf("5. Watch all customer's serial number.                     |\n");
    printf("6. Exit.                                                   |\n");
    printf("***********************************************************|\n");
    printf(">> ");
}
```

viewing customer data, adding appointments, serving appointments by serial number, checking all appointment serial numbers, and exiting the program. Users are prompted to enter a corresponding number to select the desired action. This function provides a clear and accessible interface for users to navigate through the system's functionalities.

Customer input function (its temporary function)

The input function is designed to facilitate the input of customer data. It prompts the user to enter details such as the customer's name, age, gender,

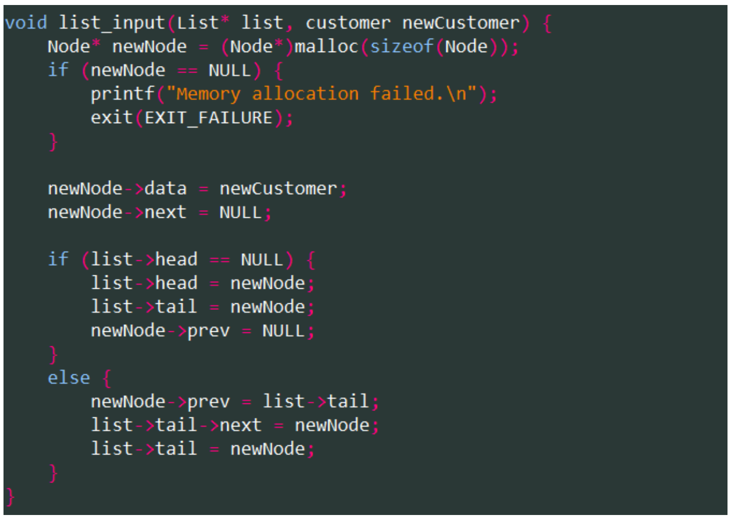


```c
customer input() {
    customer newCustomer;

    printf("Enter data for new customer:\n\n");
    printf("Customer name: ");
    scanf(" %[^\n]", newCustomer.name);
    printf("Customer age: ");
    scanf("%d", &newCustomer.age);
    printf("Customer gender: (ONLY-(M/F):");
    scanf(" %c", &newCustomer.gender);

    printf("Customer contact number(18 digit): ");
    scanf(" %[^\n]", newCustomer.contact);
    printf("Customer email: ");
    scanf(" %[^\n]", newCustomer.email);

    return newCustomer;
}
```

contact number, and email. The entered data is then stored in a customer structure, creating a cohesive unit of information for a new customer. This function encapsulates the process of gathering customer details, making the code modular and easy to understand.

List input function

The list_input function handles the insertion of a new customer into the appropriate linked list based on gender (male or female). It creates a new node

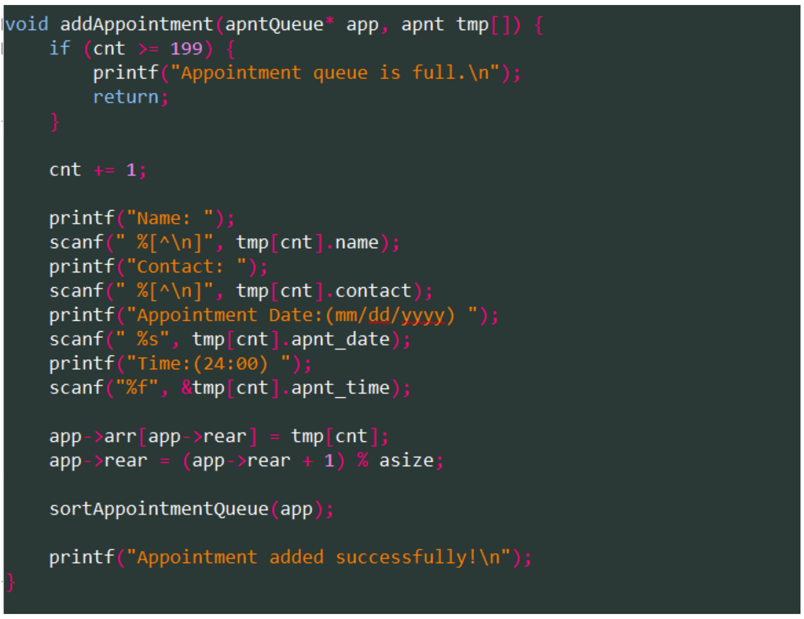


```c
void list_input(List* list, customer newCustomer) {
    Node* newNode = (Node*)malloc(sizeof(Node));
    if (newNode == NULL) {
        printf("Memory allocation failed.\n");
        exit(EXIT_FAILURE);
    }

    newNode->data = newCustomer;
    newNode->next = NULL;

    if (list->head == NULL) {
        list->head = newNode;
        list->tail = newNode;
        newNode->prev = NULL;
    }
    else {
        newNode->prev = list->tail;
        list->tail->next = newNode;
        list->tail = newNode;
    }
}
```

to store the customer data and adds it to the end of the corresponding list. The function checks whether the list is empty and appropriately updates the head and tail pointers. This function ensures that customer data is organized and stored in a linked list structure, facilitating easy retrieval and manipulation.

Add appointment function

The addAppointment function manages the process of adding a new appointment to the appointment queue. It checks if the queue is not full before prompting the user for details such as the customer's name, contact,

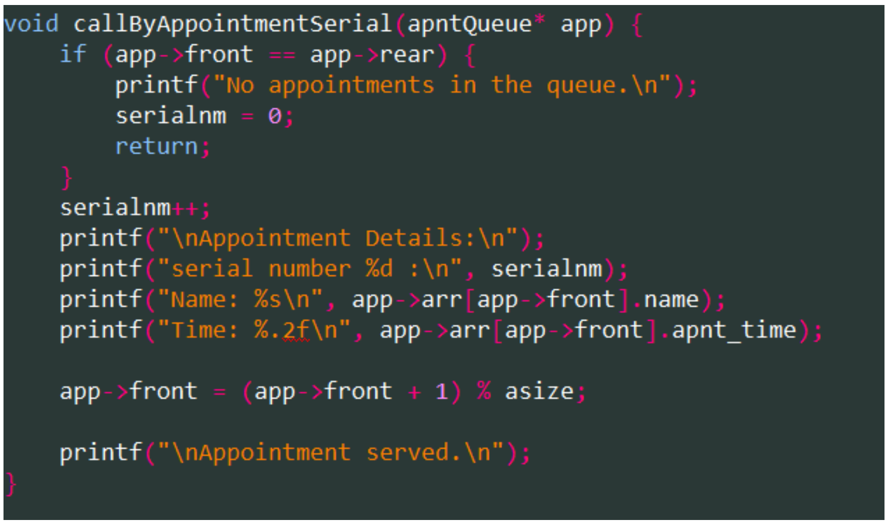


```c
void addAppointment(apntQueue* app, apnt tmp[]) {
    if (cnt >= 199) {
        printf("Appointment queue is full.\n");
        return;
    }

    cnt += 1;

    printf("Name: ");
    scanf(" %[^\n]", tmp[cnt].name);
    printf("Contact: ");
    scanf(" %[^\n]", tmp[cnt].contact);
    printf("Appointment Date:(mm/dd/yyyy) ");
    scanf(" %s", tmp[cnt].apnt_date);
    printf("Time:(24:00) ");
    scanf("%f", &tmp[cnt].apnt_time);

    app->arr[app->rear] = tmp[cnt];
    app->rear = (app->rear + 1) % asize;

    sortAppointmentQueue(app);

    printf("Appointment added successfully!\n");
}
```

appointment date, and time. The appointment data is stored in the queue, and the function then calls sortAppointmentQueue to maintain the order of appointments based on date and time. This function ensures that new appointments are seamlessly integrated into the system.

Call by appointmet function

The callByAppointmentSerial function simulates the process of serving an appointment based on its serial number. It checks if there are appointments in

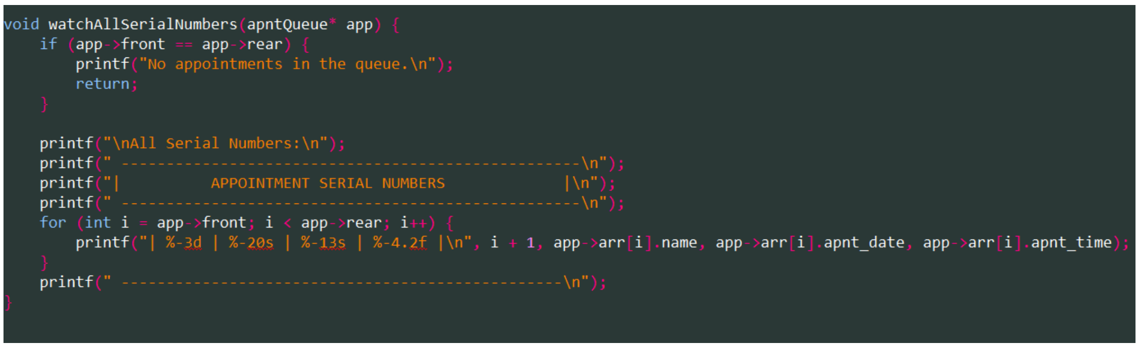


```c
void callByAppointmentSerial(apntQueue* app) {
    if (app->front == app->rear) {
        printf("No appointments in the queue.\n");
        serialnm = 0;
        return;
    }
    serialnm++;
    printf("\nAppointment Details:\n");
    printf("serial number %d :\n", serialnm);
    printf("Name: %s\n", app->arr[app->front].name);
    printf("Time: %.2f\n", app->arr[app->front].apnt_time);

    app->front = (app->front + 1) % asize;

    printf("\nAppointment served.\n");
}
```

the queue and prints details such as the customer's name and appointment time. After serving the appointment, the front index of the queue is incremented to dequeue the served appointment. This function mimics the real-world scenario of attending to customers based on their scheduled appointments.

Watch all serial appointment function

The watchAllSerialNumbers function displays all appointment serial numbers, along with the respective customer names, appointment dates, and times.

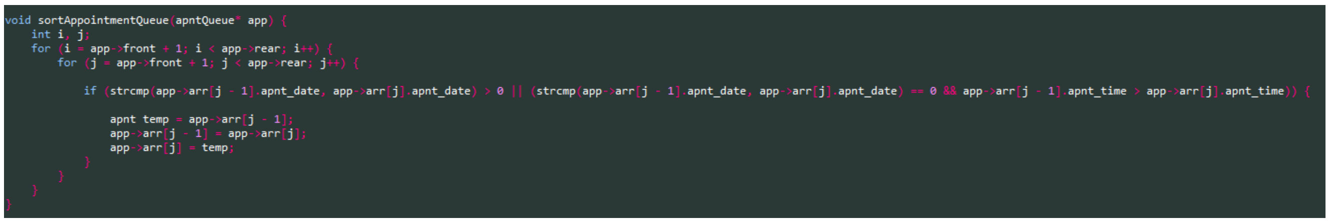


```c
void watchAllSerialNumbers(apntQueue* app) {
    if (app->front == app->rear) {
        printf("No appointments in the queue.\n");
        return;
    }

    printf("\nAll Serial Numbers:\n");
    printf(" --------------------------------------------------\n");
    printf("|          APPOINTMENT SERIAL NUMBERS              |\n");
    printf(" --------------------------------------------------\n");
    for (int i = app->front; i < app->rear; i++) {
        printf("| %-3d | %-20s | %-13s | %-4.2f |\n", i + 1, app->arr[i].name, app->arr[i].apnt_date, app->arr[i].apnt_time);
    }
    printf(" ------------------------------------------------\n");
}
```

It checks if the appointment queue is not empty before iterating through the queue, presenting a tabular overview of all scheduled appointments. This function offers users a comprehensive view of the existing appointment schedule, aiding in better management and tracking of customer appointments.

Sort function (bubble sort)

The sortAppointmentQueue function employs the bubble sort algorithm to arrange appointments in the queue in ascending order based on date and time.

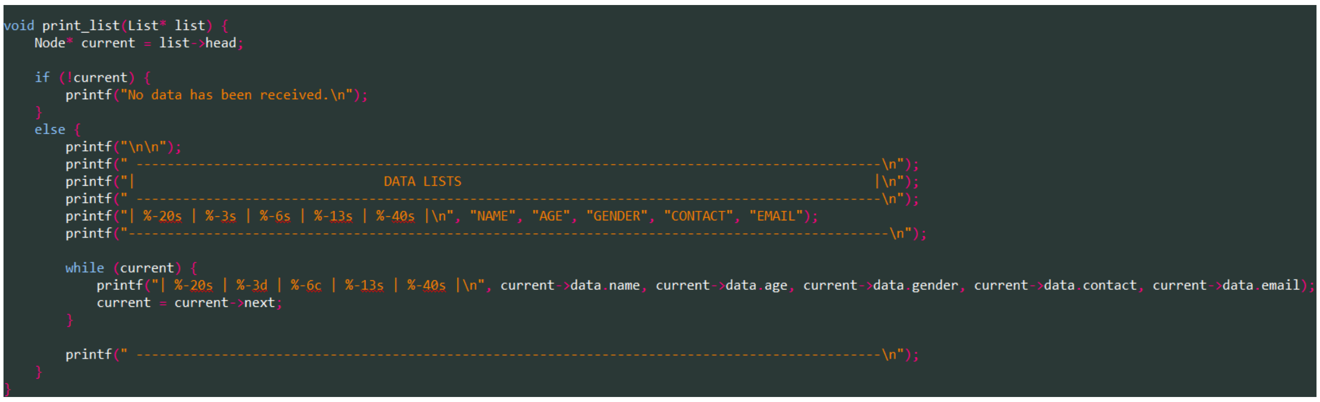


```c
void sortAppointmentQueue(apntQueue* app) {
    int i, j;
    for (i = app->front + 1; i < app->rear; i++) {
        for (j = app->front + 1; j < app->rear; j++) {

            if (strcmp(app->arr[j - 1].apnt_date, app->arr[j].apnt_date) > 0 || (strcmp(app->arr[j - 1].apnt_date, app->arr[j].apnt_date) == 0 && app->arr[j - 1].apnt_time > app->arr[j].apnt_time)) {

                apnt temp = app->arr[j - 1];
                app->arr[j - 1] = app->arr[j];
                app->arr[j] = temp;
            }
        }
    }
}
```

It iterates through the queue, comparing adjacent elements, and swaps them if they are out of order. This sorting mechanism guarantees that appointments are processed in chronological order, providing an organized and efficient approach to managing appointments.

Print_list function

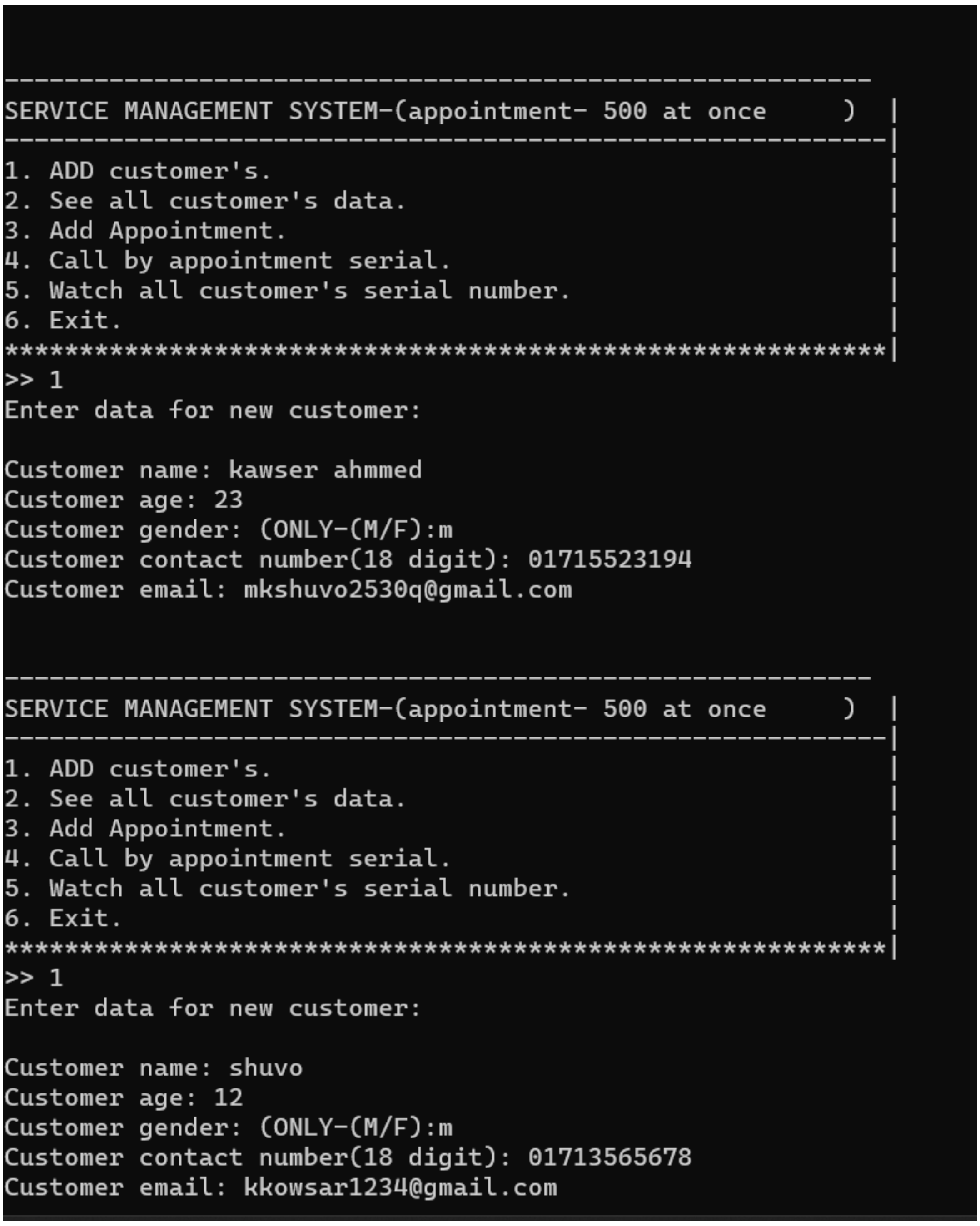
The purpose of the print_list function is to present a formatted display of customer data stored in the linked lists. By iterating through the lists and



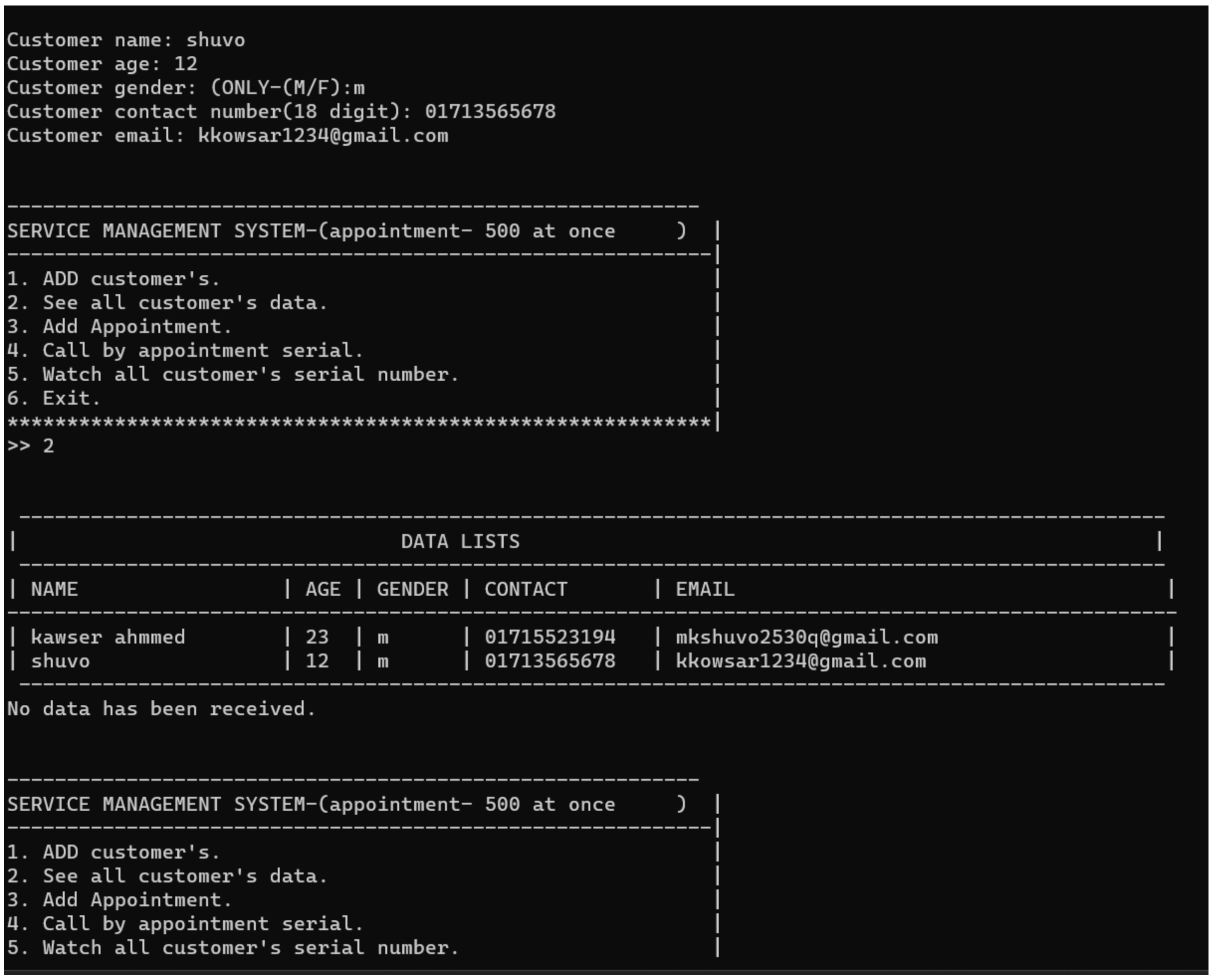
```c
void print_list(List* list) {
    Node* current = list->head;

    if (!current) {
        printf("No data has been received.\n");
    }
    else {
        printf("\n\n");
        printf(" -----------------------------------------------------------------------------------------------\n");
        printf("|                                DATA LISTS                                                     |\n");
        printf(" -----------------------------------------------------------------------------------------------\n");
        printf("| %-20s | %-3s | %-6s | %-13s | %-40s |\n", "NAME", "AGE", "GENDER", "CONTACT", "EMAIL");
        printf("------------------------------------------------------------------------------------------------\n");

        while (current) {
            printf("| %-20s | %-3d | %-6c | %-13s | %-40s |\n", current->data.name, current->data.age, current->data.gender, current->data.contact, current->data.email);
            current = current->next;
        }

        printf(" -----------------------------------------------------------------------------------------------\n");
    }
}
```

printing details such as name, age, gender, contact, and email in a tabular format, this function offers a comprehensive overview of the clientele.

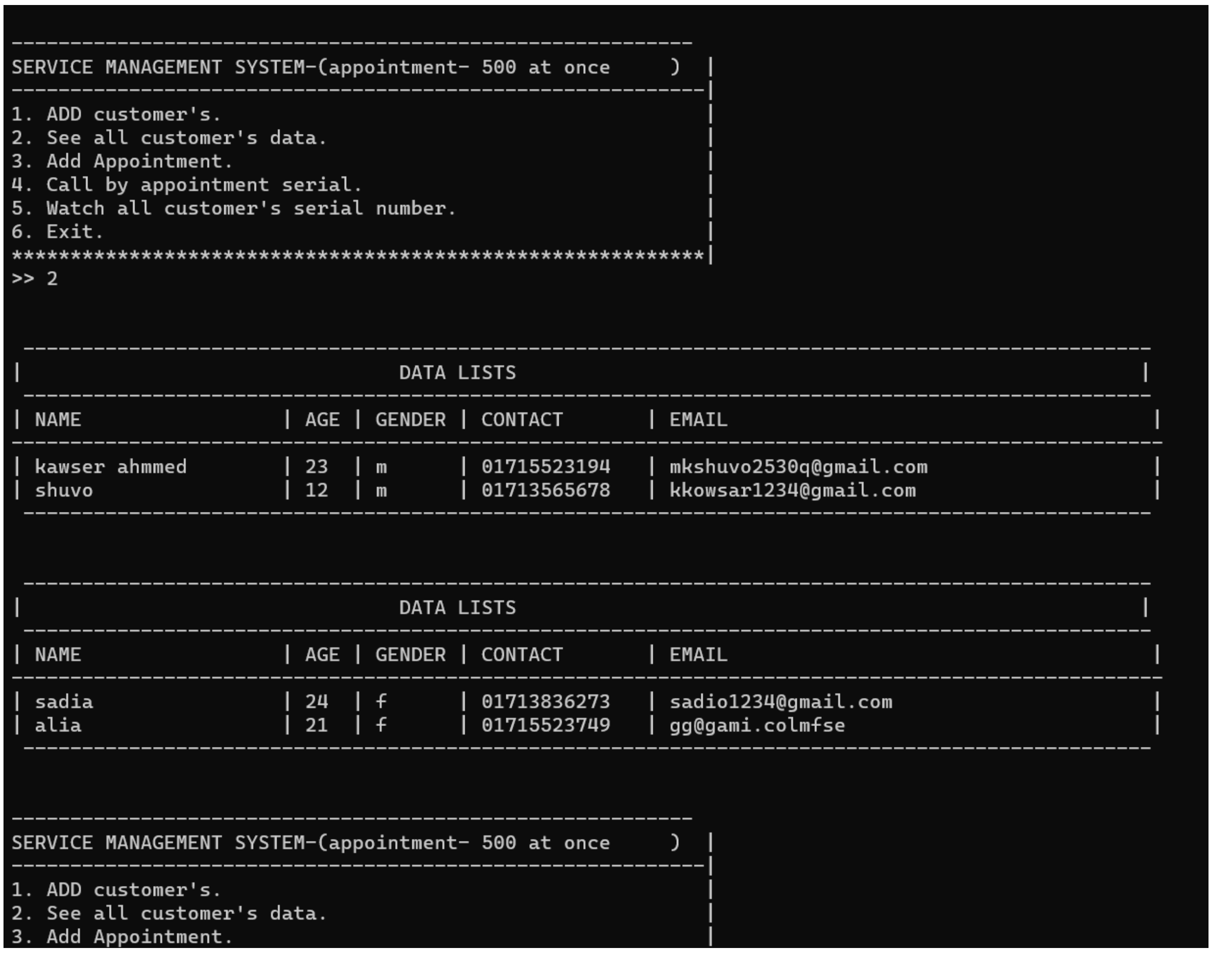
Output:



```
---------------------------------------------------------
SERVICE MANAGEMENT SYSTEM-(appointment- 500 at once     )  |
-----------------------------------------------------------|
1. ADD customer's.                                         |
2. See all customer's data.                                |
3. Add Appointment.                                        |
4. Call by appointment serial.                             |
5. Watch all customer's serial number.                     |
6. Exit.                                                   |
***********************************************************|
>> 1
Enter data for new customer:

Customer name: kawser ahmmed
Customer age: 23
Customer gender: (ONLY-(M/F):m
Customer contact number(18 digit): 01715523194
Customer email: mkshuvo2530q@gmail.com


---------------------------------------------------------
SERVICE MANAGEMENT SYSTEM-(appointment- 500 at once     )  |
-----------------------------------------------------------|
1. ADD customer's.                                         |
2. See all customer's data.                                |
3. Add Appointment.                                        |
4. Call by appointment serial.                             |
5. Watch all customer's serial number.                     |
6. Exit.                                                   |
***********************************************************|
>> 1
Enter data for new customer:

Customer name: shuvo
Customer age: 12
Customer gender: (ONLY-(M/F):m
Customer contact number(18 digit): 01713565678
Customer email: kkowsar1234@gmail.com
```



```
Customer name: shuvo
Customer age: 12
Customer gender: (ONLY-(M/F):m
Customer contact number(18 digit): 01713565678
Customer email: kkowsar1234@gmail.com


----------------------------------------------------------
SERVICE MANAGEMENT SYSTEM-(appointment- 500 at once     )  |
-----------------------------------------------------------|
1. ADD customer's.                                         |
2. See all customer's data.                                |
3. Add Appointment.                                        |
4. Call by appointment serial.                             |
5. Watch all customer's serial number.                     |
6. Exit.                                                   |
***********************************************************|
>> 2


 -------------------------------------------------------------------------------------------------
|                                DATA LISTS                                                       |
 -------------------------------------------------------------------------------------------------
| NAME                 | AGE | GENDER | CONTACT        | EMAIL                                     |
---------------------------------------------------------------------------------------------------
| kawser ahmmed        | 23  | m      | 01715523194    | mkshuvo2530q@gmail.com                    |
| shuvo                | 12  | m      | 01713565678    | kkowsar1234@gmail.com                     |
 -------------------------------------------------------------------------------------------------
No data has been received.


----------------------------------------------------------
SERVICE MANAGEMENT SYSTEM-(appointment- 500 at once     )  |
-----------------------------------------------------------|
1. ADD customer's.                                         |
2. See all customer's data.                                |
3. Add Appointment.                                        |
4. Call by appointment serial.                             |
5. Watch all customer's serial number.                     |
```



```
----------------------------------------------------------
SERVICE MANAGEMENT SYSTEM-(appointment- 500 at once     )  |
-----------------------------------------------------------|
1. ADD customer's.                                         |
2. See all customer's data.                                |
3. Add Appointment.                                        |
4. Call by appointment serial.                             |
5. Watch all customer's serial number.                     |
6. Exit.                                                   |
***********************************************************|
>> 2


 -------------------------------------------------------------------------------------------------
|                                DATA LISTS                                                       |
 -------------------------------------------------------------------------------------------------
| NAME                 | AGE | GENDER | CONTACT        | EMAIL                                     |
---------------------------------------------------------------------------------------------------
| kawser ahmmed        | 23  | m      | 01715523194    | mkshuvo2530q@gmail.com                    |
| shuvo                | 12  | m      | 01713565678    | kkowsar1234@gmail.com                     |
 -------------------------------------------------------------------------------------------------


 -------------------------------------------------------------------------------------------------
|                                DATA LISTS                                                       |
 -------------------------------------------------------------------------------------------------
| NAME                 | AGE | GENDER | CONTACT        | EMAIL                                     |
---------------------------------------------------------------------------------------------------
| sadia                | 24  | f      | 01713836273    | sadio1234@gmail.com                       |
| alia                 | 21  | f      | 01715523749    | gg@gami.colmfse                           |
 -------------------------------------------------------------------------------------------------


----------------------------------------------------------
SERVICE MANAGEMENT SYSTEM-(appointment- 500 at once     )  |
-----------------------------------------------------------|
1. ADD customer's.                                         |
2. See all customer's data.                                |
3. Add Appointment.                                        |
```