DBMS ASSIGNMENT – 1

SEQUENTIAL FILE PROCESSING

***Roll Number: U19CS012***

***Name: BHAGYA VINOD RANA***

**Code:**

*#include* <stdio.h>

*#include* <string.h>

*#include* <stdlib.h>

*// BASIC STRUCTURE OF PATIENT*

struct date

{

    int dd;

    int mm;

    int yy;

};

struct patient

{

    int pno;

    char first\_name[25];

    char last\_name[25];

    int age;

    char gender;

    char area[25];

    struct date admission;

    struct date discharge;

};

*// Global Counter for Number of Records in Database*

int record\_count = 0;

*// To Intialise Count of Records*

void init();

*// 1 - Add Record in File*

void add();

*// 2 - Deletion of Record in File*

void del();

*// 3 - Modify a Record in File*

void modify();

*// 4 - Generate Summary Report*

void summary();

*// 5 - Sort the Data*

*// 5.1 - Sort by First Name*

void sort\_first\_name(int ch);

*// 5.2 - Sort by Last Name*

void sort\_last\_name(int ch);

*// 5.3 - Sort by Age*

void sort\_age(int ch);

*// To Compare Two Dates*

int cmp(struct date d1, struct date d2);

*// 5.4 - Sort by Dates*

void sort\_date(int ch);

*// Main Function that instructs to sort based on which parameter*

void sort();

*// 6 - List all Records of File for Specific Range*

void List\_Range();

*// 7 - Seperate Record Based on Gender*

void Split\_by\_Gender();

*// 8 - To Display all The Records in File*

void display();

int main()

{

*// To Intialize the Record Count*

    init();

    printf("~~ COVID PATIENT DATA [SURAT] ~~\n");

*while* (1)

    {

        int ch;

        printf("\n1 -> Add a Record of Patient\n");

        printf("2 -> Delete a Record of Patient\n");

        printf("3 -> Modify a Record of Patient\n");

        printf("4 -> Generate Summary Report\n");

        printf("5 -> Sort the Record(s)\n");

        printf("6 -> List all Records of File for Specific Range\n");

        printf("7 -> Seperate Record Based on Gender\n");

        printf("8 -> Display a Record of All Patients\n");

        printf("9 -> Exit\n");

        scanf("%d", &ch);

*switch* (ch)

        {

*case* 1:

            add();

*break*;

*case* 2:

            del();

*break*;

*case* 3:

            modify();

*break*;

*case* 4:

            summary();

*break*;

*case* 5:

            sort();

*break*;

*case* 6:

            List\_Range();

*break*;

*case* 7:

            Split\_by\_Gender();

*break*;

*case* 8:

            display();

*break*;

*case* 9:

            exit(0);

*break*;

*default*:

            printf("Enter a Valid Choice!\n");

*break*;

        }

    }

*return* 0;

}

*// To Intialise Count of Records*

void init()

{

    FILE \*fp;

    fp = fopen("data.txt", "r+");

*if* (fp == NULL)

    {

        fclose(fp);

*return*;

    }

*else*

    {

        struct patient p;

*while* (fread(&p, sizeof(struct patient), 1, fp))

        {

            record\_count++;

        }

        fclose(fp);

*return*;

    }

}

*// ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~ADD~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~*

*// 1 - Add Record in File*

void add()

{

    FILE \*fp;

    fp = fopen("data.txt", "a+");

    struct patient p1;

    printf("Patient's Number : \n");

    scanf("%d", &p1.pno);

    printf("Patient's First Name : \n");

    fflush(stdin);

    gets(p1.first\_name);

    printf("Patient's Last Name : \n");

    fflush(stdin);

    gets(p1.last\_name);

    printf("Patient's Age : \n");

    scanf("%d", &p1.age);

    printf("Patient's Gender [M(m)/F(f)] : \n");

    fflush(stdin);

    scanf("%c", &p1.gender);

    printf("Enter Area of the patient : \n");

    fflush(stdin);

    gets(p1.area);

    printf("Patients Admission Date [DD/MM/YYYY] : \n");

    scanf("%d %d %d", &p1.admission.dd, &p1.admission.mm, &p1.admission.yy);

    printf("Patients Discharge Date [DD/MM/YYYY] :  \n");

    scanf("%d %d %d", &p1.discharge.dd, &p1.discharge.mm, &p1.discharge.yy);

*// Write the Data in File*

    fwrite(&p1, sizeof(struct patient), 1, fp);

*// Increment the Record Count*

    record\_count++;

    fclose(fp);

}

*// ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~DELETE~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~*

*// 2 - Deletion of Record in File*

void del()

{

    FILE \*fp;

    FILE \*fp1;

    fp = fopen("data.txt", "r+");

*if* (fp == NULL)

    {

        printf("Unable to Open File!\n");

        fclose(fp);

*return*;

    }

*else*

    {

        fp1 = fopen("copy.txt", "a+");

        int id, found;

        printf("\nEnter the Patient Number to be deleted : ");

        scanf("%d", &id);

        struct patient p1;

*// Write all records to the tempfile, except the one(s) you want to delete.*

*while* (fread(&p1, sizeof(struct patient), 1, fp))

        {

*if* (p1.pno != id)

            {

                fwrite(&p1, sizeof(struct patient), 1, fp1);

            }

*else*

            {

                found = 1;

            }

        }

        fclose(fp);

        fclose(fp1);

*// Remove the old file.*

        remove("data.txt");

*// Rename New File With Old File Name*

        rename("copy.txt", "data.txt");

*if* (found == 1)

        {

            printf("\nRecord Deleted Succesfully!\n");

            record\_count -= 1; *// 1 Record Deleted*

        }

*else*

        {

            printf("\nRecord Not Found in DataBase![Already Deleted/Not Added]!\n");

        }

    }

}

*// ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~MODIFY~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~*

*// 3 - Modify a Record in File*

void modify()

{

    FILE \*fp;

    FILE \*fp1;

    fp = fopen("data.txt", "r+");

*if* (fp == NULL)

    {

        printf("Unable to Open File!\n");

        fclose(fp);

*return*;

    }

*else*

    {

        struct patient p1, p2;

        int found = 0, id;

        fp1 = fopen("copy.txt", "a+");

*// New Record*

        printf("\nEnter the Patient Number to be updated : ");

        scanf("%d", &id);

        p1.pno = id;

        printf("Enter First Name of patient : ");

        fflush(stdin);

        gets(p1.first\_name);

        printf("Enter Last Name of patient : ");

        fflush(stdin);

        gets(p1.last\_name);

        printf("Enter age of patient : ");

        scanf("%d", &p1.age);

        printf("Enter gender of patient (M/F) : ");

        fflush(stdin);

        scanf("%c", &p1.gender);

        printf("Enter Area of the patient : ");

        fflush(stdin);

        gets(p1.area);

        printf("Enter Admission date (format : dd mm yyyy ) : ");

        scanf("%d %d %d", &p1.admission.dd, &p1.admission.mm, &p1.admission.yy);

        printf("Enter Discharge date (format : dd mm yyyy ) : ");

        scanf("%d %d %d", &p1.discharge.dd, &p1.discharge.mm, &p1.discharge.yy);

*while* (fread(&p2, sizeof(struct patient), 1, fp))

        {

*if* (p2.pno != id)

            {

                fwrite(&p2, sizeof(struct patient), 1, fp1);

            }

*else*

            {

*// Write the New Data inplace of Old One*

                fwrite(&p1, sizeof(struct patient), 1, fp1);

                found = 1;

            }

        }

        fclose(fp);

        fclose(fp1);

*// Remove the old file.*

        remove("data.txt");

*// Rename New File With Old File Name*

        rename("copy.txt", "data.txt");

*if* (found == 1)

        {

            printf("\nRecord Modified Succesfully!\n");

        }

*else*

        {

            printf("\nRecord Not Found in the DataBase!\n");

        }

    }

}

*// ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~SUMMARY~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~*

*// 4 - Generate Summary Report*

void summary()

{

    FILE \*fp;

    fp = fopen("data.txt", "r+");

*if* (fp == NULL)

    {

        printf("Unable to Open File!\n");

        fclose(fp);

*return*;

    }

    struct patient p1, p2;

*// Array to Store Age's Frequency [Max Human Age = 150]*

    int age[150] = {0};

    int male = 0;

    int female = 0;

*while* (fread(&p1, sizeof(struct patient), 1, fp))

    {

        age[p1.age]++;

        char ch = p1.gender;

*if* (ch == 'm' || ch == 'M')

        {

            male += 1;

        }

*else*

        {

            female += 1;

        }

    }

    fclose(fp);

    printf("\n~~ SUMMARY ~~\n\n");

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

    printf("Number of Patient's Record in DataBase = %d\n", record\_count);

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

    printf("Number of Male Patient's Record in DataBase   = %d\n", male);

    printf("Number of Female Patient's Record in DataBase = %d\n", female);

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

    printf("Age   |   Number of Patients\n");

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

*for* (int i = 1; i < 150; i++)

    {

*if* (age[i] != 0)

            printf("%d              %d\n", i, age[i]);

    }

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

    printf("Area     |      Number of Patients\n");

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

*// v = Iterator of Visited Array*

    int freq, flag = 0, v = 0;

    fp = fopen("data.txt", "r+");

    char word[25], visited[record\_count][25];

*while* (fread(&p1, sizeof(struct patient), 1, fp))

    {

        flag = 0;

        strcpy(word, p1.area);

*for* (int i = 0; i < v; i++)

        {

*if* (strcmp(word, visited[i]) == 0)

                flag = 1;

        }

*// Not A Unique Area*

*if* (flag == 1)

        {

*// Frequency already counted*

*continue*;

        }

*else*

        {

*// Insert Unique Area*

            strcpy(visited[v], word);

            v++;

        }

        FILE \*fp1;

        freq = 0;

        fp1 = fopen("data.txt", "r+");

*while* (fread(&p2, sizeof(struct patient), 1, fp1))

        {

*if* (strcmp(word, p2.area) == 0)

                freq++;

        }

        printf("%s     :     %d \n", word, freq);

        fclose(fp1);

    }

    fclose(fp);

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

}

*// ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~SORTING~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~*

*// 5.1 - Sort by First Name*

void sort\_first\_name(int ch)

{

    struct patient p[record\_count], p1, temp;

    int i = 0, j;

    FILE \*fp;

    fp = fopen("data.txt", "r+");

*while* (fread(&p1, sizeof(struct patient), 1, fp))

    {

        p[i] = p1;

        i++;

    }

*// Classical Bubble Sort to Sort the Array*

*// Ascending*

*if* (ch == 1)

    {

*for* (i = 1; i < record\_count; i++)

        {

*for* (j = 0; j < record\_count - i; j++)

            {

*if* (strcmp(p[j + 1].first\_name, p[j].first\_name) < 0)

                {

                    temp = p[j];

                    p[j] = p[j + 1];

                    p[j + 1] = temp;

                }

            }

        }

    }

*else*

    { *// Descending*

*for* (i = 1; i < record\_count; i++)

        {

*for* (j = 0; j < record\_count - i; j++)

            {

*if* (strcmp(p[j + 1].first\_name, p[j].first\_name) > 0)

                {

                    temp = p[j];

                    p[j] = p[j + 1];

                    p[j + 1] = temp;

                }

            }

        }

    }

    FILE \*fp1;

    fp1 = fopen("sorted.txt", "a+");

*for* (i = 0; i < record\_count; i++)

    {

        fwrite(&p[i], sizeof(struct patient), 1, fp1);

    }

    fclose(fp);

    fclose(fp1);

    remove("data.txt");

    rename("sorted.txt", "data.txt");

    printf("\n\nSorted Record(s) are : \n");

    display();

}

*// 5.2 - Sort by Last Name*

void sort\_last\_name(int ch)

{

    struct patient p[record\_count], p1, temp;

    int i = 0, j;

    FILE \*fp;

    fp = fopen("data.txt", "r+");

*while* (fread(&p1, sizeof(struct patient), 1, fp))

    {

        p[i] = p1;

        i++;

    }

*// Ascending*

*if* (ch == 1)

    {

*for* (i = 1; i < record\_count; i++)

        {

*for* (j = 0; j < record\_count - i; j++)

            {

*if* (strcmp(p[j + 1].last\_name, p[j].last\_name) < 0)

                {

                    temp = p[j];

                    p[j] = p[j + 1];

                    p[j + 1] = temp;

                }

            }

        }

    }

*else*

    { *// Descending*

*for* (i = 1; i < record\_count; i++)

        {

*for* (j = 0; j < record\_count - i; j++)

            {

*if* (strcmp(p[j + 1].last\_name, p[j].last\_name) > 0)

                {

                    temp = p[j];

                    p[j] = p[j + 1];

                    p[j + 1] = temp;

                }

            }

        }

    }

    FILE \*fp1;

    fp1 = fopen("sorted.txt", "a+");

*for* (i = 0; i < record\_count; i++)

    {

        fwrite(&p[i], sizeof(struct patient), 1, fp1);

    }

    fclose(fp);

    fclose(fp1);

    remove("data.txt");

    rename("sorted.txt", "data.txt");

    printf("\n\nSorted Record(s) are : \n");

    display();

}

*// 5.3 - Sort by Age*

void sort\_age(int ch)

{

    struct patient p[record\_count], p1, temp;

    int i = 0, j;

    FILE \*fp;

    fp = fopen("data.txt", "r+");

*while* (fread(&p1, sizeof(struct patient), 1, fp))

    {

        p[i] = p1;

        i++;

    }

*// Ascending*

*if* (ch == 1)

    {

*for* (i = 1; i < record\_count; i++)

        {

*for* (j = 0; j < record\_count - i; j++)

            {

*if* (p[j + 1].age < p[j].age)

                {

                    temp = p[j];

                    p[j] = p[j + 1];

                    p[j + 1] = temp;

                }

            }

        }

    }

*else*

    { *// Descending*

*for* (i = 1; i < record\_count; i++)

*for* (j = 0; j < record\_count - i; j++)

            {

*if* (p[j + 1].age > p[j].age)

                {

                    temp = p[j];

                    p[j] = p[j + 1];

                    p[j + 1] = temp;

                }

            }

    }

    FILE \*fp1;

    fp1 = fopen("sorted.txt", "a+");

*for* (i = 0; i < record\_count; i++)

    {

        fwrite(&p[i], sizeof(struct patient), 1, fp1);

    }

    fclose(fp);

    fclose(fp1);

    remove("data.txt");

    rename("sorted.txt", "data.txt");

    printf("\n\nSorted Record is : \n");

    display();

}

*// To Compare Two Dates*

int cmp(struct date d1, struct date d2)

{

*// Year's*

*if* (d1.yy > d2.yy)

*return* 1;

*else* *if* (d1.yy < d2.yy)

*return* -1;

*else*

    {

*// Month's*

*if* (d1.mm > d2.mm)

*return* 1;

*else* *if* (d1.mm < d2.mm)

*return* -1;

*else*

        {

*// Date*

*if* (d1.dd > d2.dd)

*return* 1;

*else* *if* (d1.dd < d2.dd)

*return* -1;

*else*

            {

*// Both are same Date*

*return* 0;

            }

        }

    }

}

*// 5.4 - Sort by Dates*

void sort\_date(int ch)

{

    struct patient p[record\_count], p1, temp;

    int i = 0, j;

    FILE \*fp;

    fp = fopen("data.txt", "r+");

*while* (fread(&p1, sizeof(struct patient), 1, fp))

    {

        p[i] = p1;

        i++;

    }

*// Ascending*

*if* (ch == 1)

    {

*for* (i = 1; i < record\_count; i++)

        {

*for* (j = 0; j < record\_count - i; j++)

            {

*if* (cmp(p[j + 1].admission, p[j].admission) < 0)

                {

                    temp = p[j];

                    p[j] = p[j + 1];

                    p[j + 1] = temp;

                }

            }

        }

    }

*else*

    { *// Desecending*

*for* (i = 1; i < record\_count; i++)

*for* (j = 0; j < record\_count - i; j++)

            {

*if* (cmp(p[j + 1].discharge, p[j].discharge) > 0)

                {

                    temp = p[j];

                    p[j] = p[j + 1];

                    p[j + 1] = temp;

                }

            }

    }

    FILE \*fp1;

    fp1 = fopen("sorted.txt", "a+");

*for* (i = 0; i < record\_count; i++)

    {

        fwrite(&p[i], sizeof(struct patient), 1, fp1);

    }

    fclose(fp);

    fclose(fp1);

    remove("data.txt");

    rename("sorted.txt", "data.txt");

    printf("\n\nSorted Record(s) are : \n");

    display();

}

*// 5 - Sort the Data*

void sort()

{

    FILE \*fp;

    fp = fopen("data.txt", "r+");

*// Check File is Able to be Opened*

*if* (fp == NULL)

    {

        printf("Unable to Open File!\n");

        fclose(fp);

*return*;

    }

    fclose(fp);

    printf("~~~~~~~SORTING RECORDS~~~~~~~\n");

    printf("1 -> Ascending Order\n");

    printf("2 -> Descending order\n");

    printf("Choice [1/2] : ");

    int ch;

    scanf("%d", &ch);

*// Ascending Order*

*if* (ch == 1)

    {

        int param; *// parameter*

        printf("\nSort By :");

        printf("\n1 -> First Name");

        printf("\n2 -> Last Name");

        printf("\n3 -> Age");

        printf("\n4 -> Admission Date");

        printf("\nChoice [1/2/3/4] : ");

        scanf("%d", &param);

*switch* (param)

        {

*case* 1:

            sort\_first\_name(ch);

*break*;

*case* 2:

            sort\_last\_name(ch);

*break*;

*case* 3:

            sort\_age(ch);

*break*;

*case* 4:

            sort\_date(ch);

*break*;

*default*:

            printf("\nInvalid choice!");

*break*;

        }

    }

*else* *if* (ch == 2)

    {

        int param; *// parameter*

        printf("\nSort by ");

        printf("\n1 -> First Name");

        printf("\n2 -> Last Name");

        printf("\n3 -> Age");

        printf("\n4 -> Discharge Date");

        printf("\nChoice [1/2/3/4] : ");

        scanf("%d", &param);

*switch* (param)

        {

*case* 1:

            sort\_first\_name(ch);

*break*;

*case* 2:

            sort\_last\_name(ch);

*break*;

*case* 3:

            sort\_age(ch);

*break*;

*case* 4:

            sort\_date(ch);

*break*;

*default*:

            printf("\nInvalid choice!");

*break*;

        }

    }

*else*

    {

        printf("\n\nInvalid Choice Entered!");

    }

}

*// ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~RANGE~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~*

*// 6 - List all Records of File for Specific Range*

void List\_Range()

{

    struct patient p;

    struct date d[11];

    int mon, year, found = 0;

    FILE \*fp;

    fp = fopen("data.txt", "r+");

*if* (fp == NULL)

    {

        printf("Unable to Open File!\n");

        fclose(fp);

*return*;

    }

    printf("~~~~~ LIST PATIENTS BASED ON RANGE ~~~~~\n");

    printf("1 -> List all Patients from B-K [B=BHAGYA]\n");

    printf("2 -> List all Patients whose Adm. Date btwn 'given date' till 10 days next\n");

    printf("3 -> List all Patients whose Adm. month btwn 'given month' till 5 months next\n");

    int ch;

    printf("\nChoice [1/2/3] : ");

    scanf("%d", &ch);

*switch* (ch)

    {

*case* 1:

*while* (fread(&p, sizeof(struct patient), 1, fp))

        {

*if* ((p.first\_name[0] >= 'B' && p.first\_name[0] <= 'K') || (p.first\_name[0] >= 'b' && p.first\_name[0] <= 'k'))

            {

                printf("\n~~ PATIENT DETAILS ~~\n");

                printf("\nPatient Number : %d", p.pno);

                printf("\nFirst Name     : %s", p.first\_name);

                printf("\nLast Name      : %s", p.last\_name);

                printf("\nAge            : %d", p.age);

                printf("\nGender         : %c", p.gender);

                printf("\nArea           : %s", p.area);

                printf("\nAdmission Date : %d / %d / %d", p.admission.dd, p.admission.mm, p.admission.yy);

                printf("\nDischarge Date : %d / %d / %d", p.discharge.dd, p.discharge.mm, p.discharge.yy);

                found = 1;

            }

        }

        fclose(fp);

*if* (found == 0)

        {

            printf("\n\nNo Records Founds for Above Input!");

        }

*break*;

*case* 2:

        printf("Enter given date [DD/MM/YYYY] : ");

        scanf("%d %d %d", &d[0].dd, &d[0].mm, &d[0].yy);

*//m -> MAXIMUM Days of Sspecific Month*

        int i = 1, m;

*// Calculate Days on Specific Month*

*while* (i != 11)

        {

*if* ((d[i - 1].mm % 2 != 0 && d[i - 1].mm <= 7) || (d[i - 1].mm % 2 == 0 && d[i - 1].mm <= 12))

            {

                m = 31;

            }

*else* *if* (d[i - 1].mm == 2)

            {

*if* (d[i - 1].yy % 400 == 0)

                    m = 29;

*else* *if* (d[i - 1].yy % 100 == 0)

                    m = 28;

*else* *if* (d[i - 1].yy % 4 == 0)

                    m = 29;

*else*

                {

                    m = 28;

                }

            }

*else*

            {

                m = 30;

            }

*if* (d[i - 1].dd == m)

            {

                d[i].dd = (d[i - 1].dd + 1) % m;

*if* (d[i - 1].mm == 12)

                {

                    d[i].mm = 1;

                    d[i].yy = d[i - 1].yy + 1;

                }

*else*

                {

                    d[i].mm = d[i - 1].mm + 1;

                    d[i].yy = d[i - 1].yy;

                }

            }

*else*

            {

                d[i].dd = (d[i - 1].dd + 1);

                d[i].mm = d[i - 1].mm;

                d[i].yy = d[i - 1].yy;

            }

            i++;

        }

*while* (fread(&p, sizeof(struct patient), 1, fp))

        {

            i = 0;

*while* (i < 11)

            {

*if* (cmp(p.admission, d[i]) == 0) *//cmp() Returns 0 -> Equal*

                {

                    printf("\n~~ PATIENT DETAILS ~~\n");

                    printf("\nPatient Number : %d", p.pno);

                    printf("\nFirst Name     : %s", p.first\_name);

                    printf("\nLast Name      : %s", p.last\_name);

                    printf("\nAge            : %d", p.age);

                    printf("\nGender         : %c", p.gender);

                    printf("\nArea           : %s", p.area);

                    printf("\nAdmission Date : %d / %d / %d", p.admission.dd, p.admission.mm, p.admission.yy);

                    printf("\nDischarge Date : %d / %d / %d", p.discharge.dd, p.discharge.mm, p.discharge.yy);

                    found = 1;

*break*;

                }

                i++;

            }

        }

        fclose(fp);

*if* (found == 0)

        {

            printf("\n\nNo Records Founds for Above Input!");

        }

*break*;

*case* 3:

        printf("\nEnter Admission Month & Year [MM/YYYY]: ");

        scanf("%d %d", &mon, &year);

*while* (fread(&p, sizeof(struct patient), 1, fp))

        {

*if* (p.admission.yy == year)

            {

*if* (mon <= 7)

                {

*if* (p.admission.mm >= mon && p.admission.mm <= mon + 5)

                    {

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

                        printf("\nPatient Number : %d", p.pno);

                        printf("\nFirst Name     : %s", p.first\_name);

                        printf("\nLast Name      : %s", p.last\_name);

                        printf("\nAge            : %d", p.age);

                        printf("\nGender         : %c", p.gender);

                        printf("\nArea           : %s", p.area);

                        printf("\nAdmission Date : %d / %d / %d", p.admission.dd, p.admission.mm, p.admission.yy);

                        printf("\nDischarge Date : %d / %d / %d", p.discharge.dd, p.discharge.mm, p.discharge.yy);

                        found = 1;

                    }

                }

*else*

                {

*if* (p.admission.mm >= mon && p.admission.mm <= 12)

                    {

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

                        printf("\nPatient Number : %d", p.pno);

                        printf("\nFirst Name     : %s", p.first\_name);

                        printf("\nLast Name      : %s", p.last\_name);

                        printf("\nAge            : %d", p.age);

                        printf("\nGender         : %c", p.gender);

                        printf("\nArea           : %s", p.area);

                        printf("\nAdmission Date : %d / %d / %d", p.admission.dd, p.admission.mm, p.admission.yy);

                        printf("\nDischarge Date : %d / %d / %d", p.discharge.dd, p.discharge.mm, p.discharge.yy);

                        found = 1;

                    }

                }

            }

*else* *if* (p.admission.yy == year + 1)

            {

*if* (p.admission.mm <= mon - 7)

                {

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

                    printf("\nPatient Number : %d", p.pno);

                    printf("\nFirst Name     : %s", p.first\_name);

                    printf("\nLast Name      : %s", p.last\_name);

                    printf("\nAge            : %d", p.age);

                    printf("\nGender         : %c", p.gender);

                    printf("\nArea           : %s", p.area);

                    printf("\nAdmission Date : %d / %d / %d", p.admission.dd, p.admission.mm, p.admission.yy);

                    printf("\nDischarge Date : %d / %d / %d", p.discharge.dd, p.discharge.mm, p.discharge.yy);

                    found = 1;

                }

            }

        }

        fclose(fp);

*if* (found == 0)

        {

            printf("\n\nNo desired Record Found!");

        }

*break*;

*default*:

        printf("\n\nInvalid Input!");

        fclose(fp);

*break*;

    }

}

*// ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~GENDER~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~*

*// 7 - Seperate Record Based on Gender*

void Split\_by\_Gender()

{

    struct patient p;

    FILE \*fp;

    FILE \*fpm; *// Male File Pointer*

    FILE \*fpf; *// Female File Pointer*

    fp = fopen("data.txt", "r+");

*if* (fp == NULL)

    {

        printf("Unable to Open File!\n");

        fclose(fp);

*return*;

    }

    fpm = fopen("maledata.txt", "a+");

    fpf = fopen("femaledata.txt", "a+");

*// Incase Text File is Not Made*

*if* (fpm != NULL)

    {

        fclose(fpm);

        remove("maledata.txt");

        fpm = fopen("maledata.txt", "a+");

    }

*if* (fpf != NULL)

    {

        fclose(fpf);

        remove("femaledata.txt");

        fpf = fopen("femaledata.txt", "a+");

    }

*while* (fread(&p, sizeof(struct patient), 1, fp))

    {

*if* (p.gender == 'M' || p.gender == 'm')

        {

            fwrite(&p, sizeof(struct patient), 1, fpm);

        }

*else* *if* (p.gender == 'F' || p.gender == 'f')

        {

            fwrite(&p, sizeof(struct patient), 1, fpf);

        }

    }

    fclose(fp);

    fclose(fpm);

    fclose(fpf);

    fpm = fopen("maledata.txt", "r+");

    fpf = fopen("femaledata.txt", "r+");

    int found = 0;

    printf("\n\n~~~~~~~ MALE PATIENTS RECORDS ~~~~~~~ ");

*while* (fread(&p, sizeof(struct patient), 1, fpm))

    {

        printf("\n~~~ PATIENT DETAILS ~~~\n");

        printf("\nPatient Number : %d", p.pno);

        printf("\nFirst Name     : %s", p.first\_name);

        printf("\nLast Name      : %s", p.last\_name);

        printf("\nAge            : %d", p.age);

        printf("\nGender         : %c", p.gender);

        printf("\nArea           : %s", p.area);

        printf("\nAdmission Date : %d / %d / %d", p.admission.dd, p.admission.mm, p.admission.yy);

        printf("\nDischarge Date : %d / %d / %d", p.discharge.dd, p.discharge.mm, p.discharge.yy);

        found = 1;

    }

*if* (found == 0)

    {

        printf("\n\nNo Male records Found!\n");

    }

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

    found = 0;

    fclose(fpm);

    printf("\n\n~~~~~~~ FEMALE PATIENTS RECORDS ~~~~~~~ ");

*while* (fread(&p, sizeof(struct patient), 1, fpf))

    {

        printf("\n~~~ PATIENT DETAILS ~~~\n");

        printf("\nPatient Number : %d", p.pno);

        printf("\nFirst Name     : %s", p.first\_name);

        printf("\nLast Name      : %s", p.last\_name);

        printf("\nAge            : %d", p.age);

        printf("\nGender         : %c", p.gender);

        printf("\nArea           : %s", p.area);

        printf("\nAdmission Date : %d / %d / %d", p.admission.dd, p.admission.mm, p.admission.yy);

        printf("\nDischarge Date : %d / %d / %d", p.discharge.dd, p.discharge.mm, p.discharge.yy);

        found = 1;

    }

*if* (found == 0)

    {

        printf("\n\nNo Female Record's Found!\n");

    }

    fclose(fpf);

}

*// ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~DISPLAY~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~*

*// 8 - To Display all The Records in File*

void display()

{

    FILE \*fp;

    fp = fopen("data.txt", "r+");

*if* (fp == NULL)

    {

        printf("Unable to Open File\n");

    }

*else*

    {

*if* (record\_count == 0)

        {

            printf("\nNo Records Found!\n");

            fclose(fp);

*return*;

        }

        struct patient p1;

*while* (fread(&p1, sizeof(struct patient), 1, fp))

        {

            printf("\n~~~ PATIENT DETAILS ~~~\n");

            printf("\nPatient Number : %d", p1.pno);

            printf("\nFirst Name     : %s", p1.first\_name);

            printf("\nLast Name      : %s", p1.last\_name);

            printf("\nAge            : %d", p1.age);

            printf("\nGender         : %c", p1.gender);

            printf("\nArea           : %s", p1.area);

            printf("\nAdmission Date : %d / %d / %d", p1.admission.dd, p1.admission.mm, p1.admission.yy);

            printf("\nDischarge Date : %d / %d / %d", p1.discharge.dd, p1.discharge.mm, p1.discharge.yy);

        }

    }

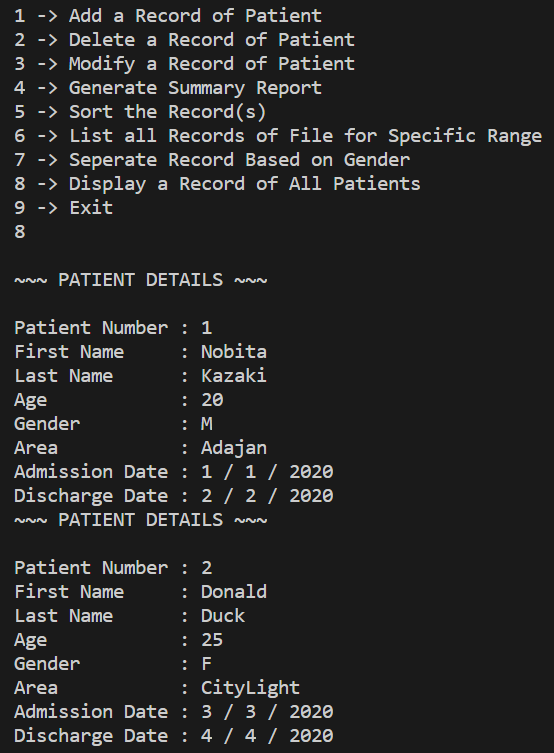
    fclose(fp);

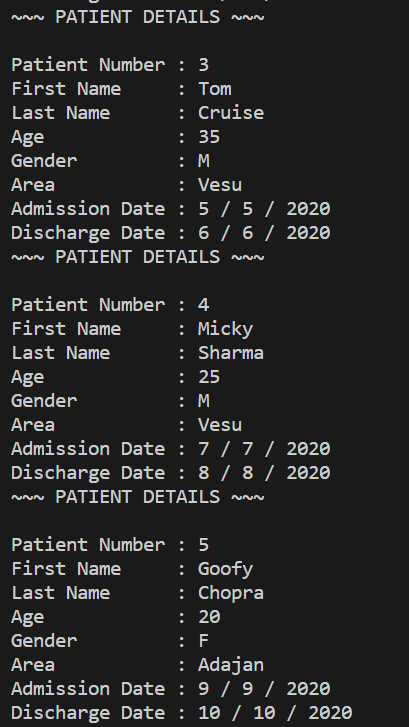
}

**Screenshots:**

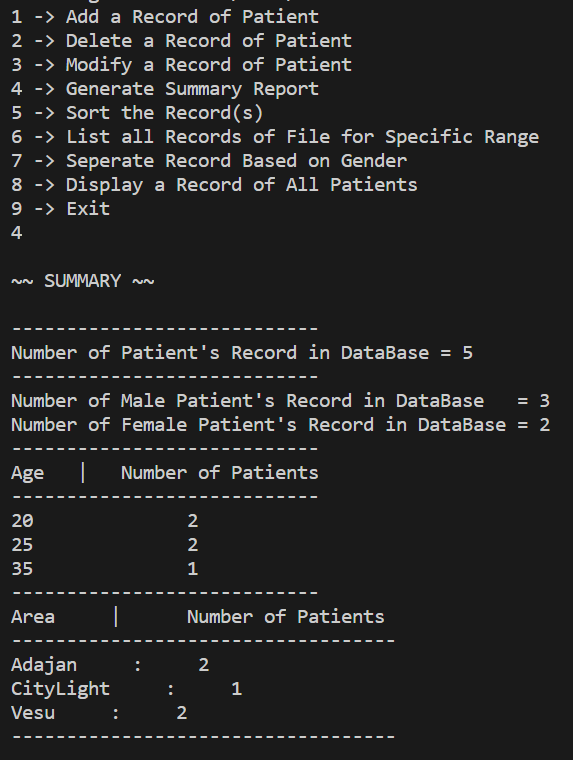
**1) Add Data of 5 Patients**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Patient No | First Name | Last Name | Age | Gender  [M/F] | Area | Admission Date | Discharge Date |
| 1 | Nobita | Kazaki | 20 | M | Adajan | 1/1/2020 | 2/2/2020 |
| 2 | Donald | Duck | 25 | F | CityLight | 3/3/2020 | 4/4/2020 |
| 3 | Tom | Cruise | 35 | M | Vesu | 5/5/2020 | 6/6/2020 |
| 4 | Micky | Sharma | 25 | M | Vesu | 7/7/2020 | 8/8/2020 |
| 5 | Goofy | Chopra | 20 | F | Adajan | 9/9/2020 | 10/10/2020 |



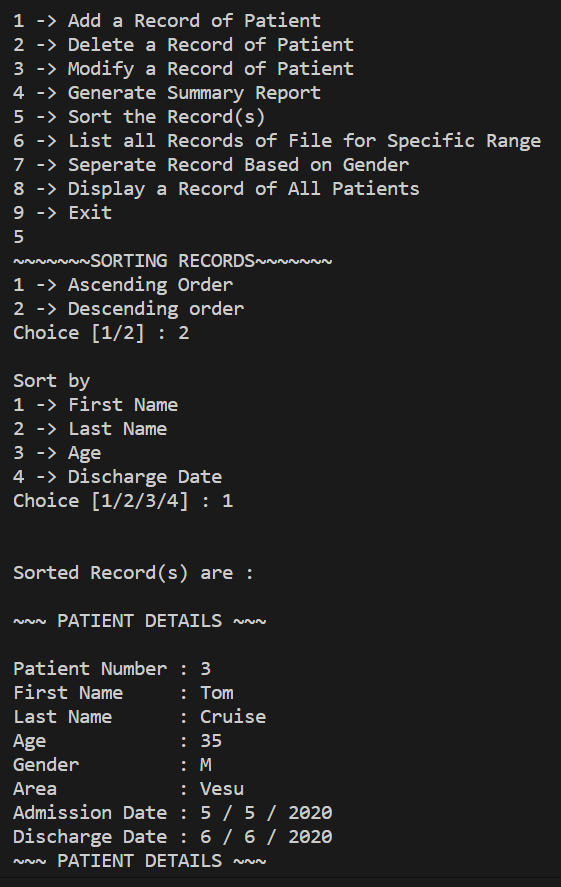


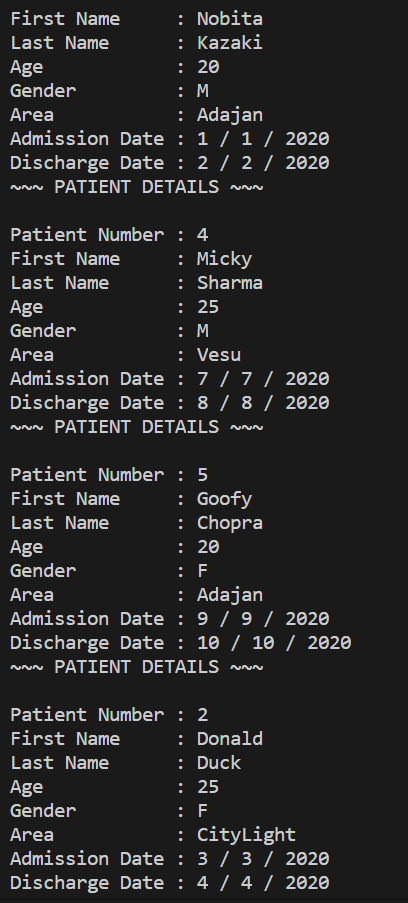
**2) Summary Reports**



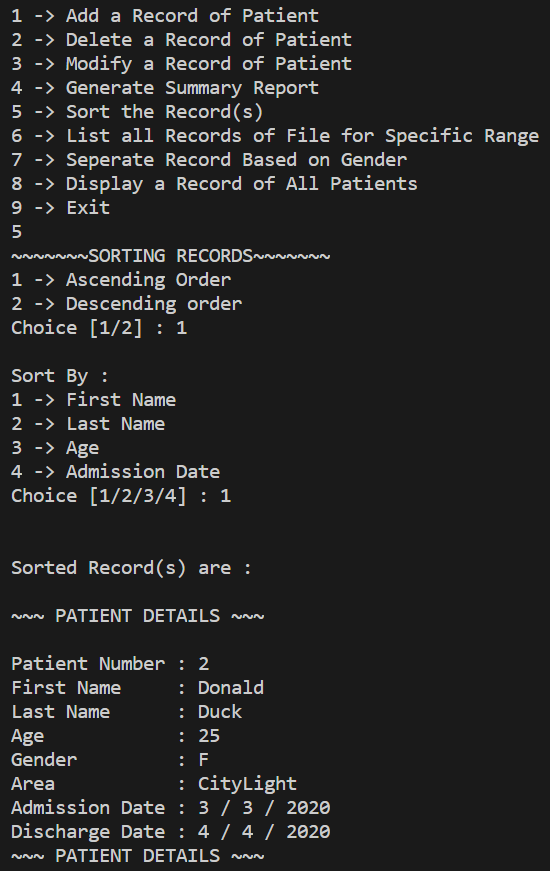
**3) Sorting Data [Ascending / Descending]**

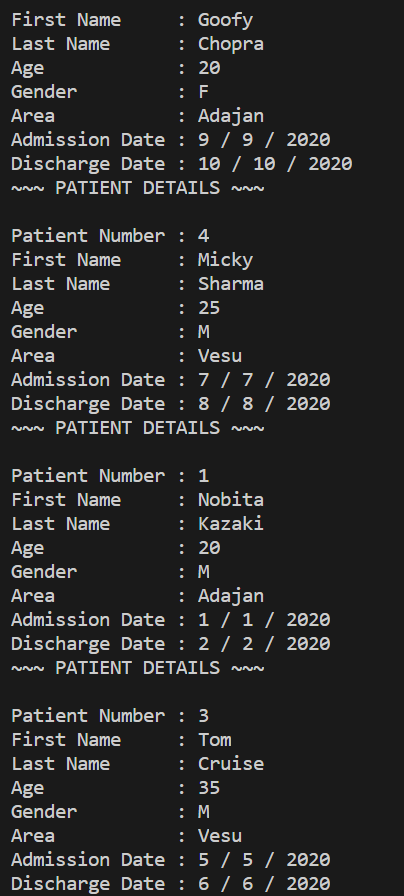
**A) Sorting in Descending Order Based on First Name**



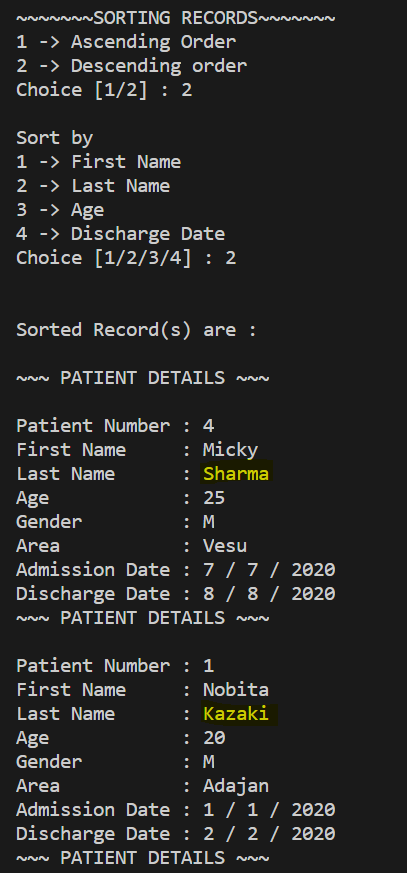


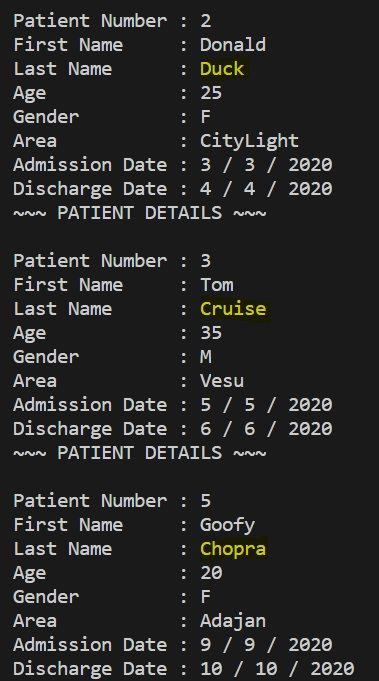
**B) Sorting in Asending Order Based on First Name**



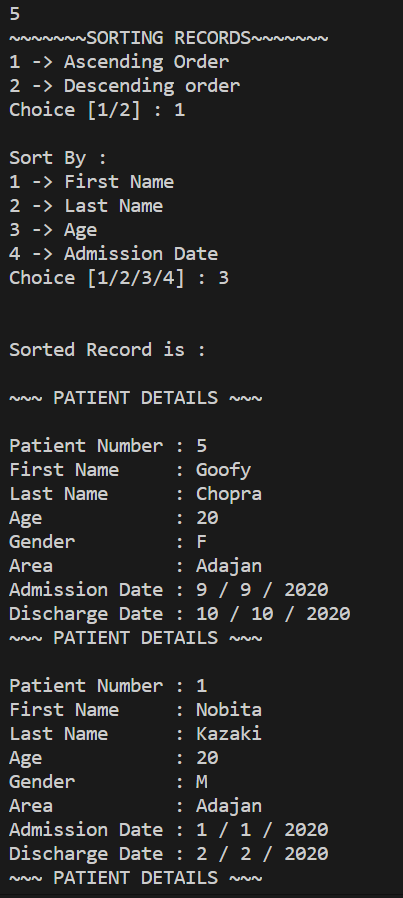


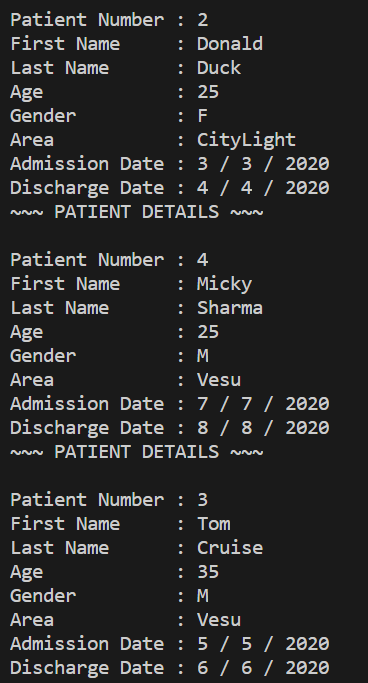
**C) Sorting in Descending Order Based on Last Name**



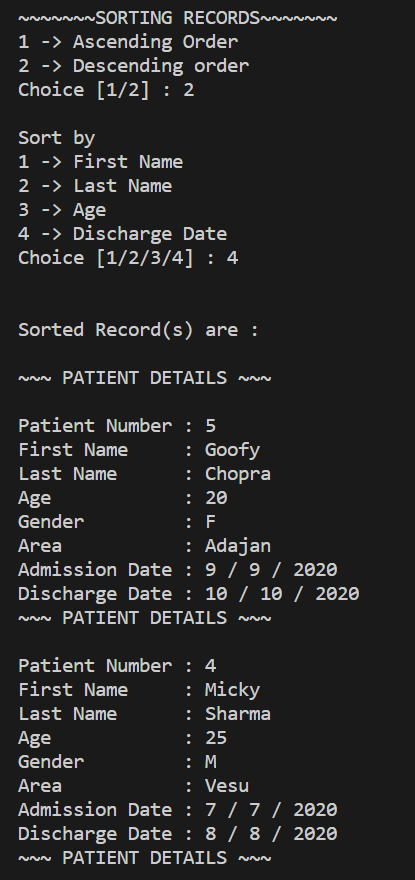


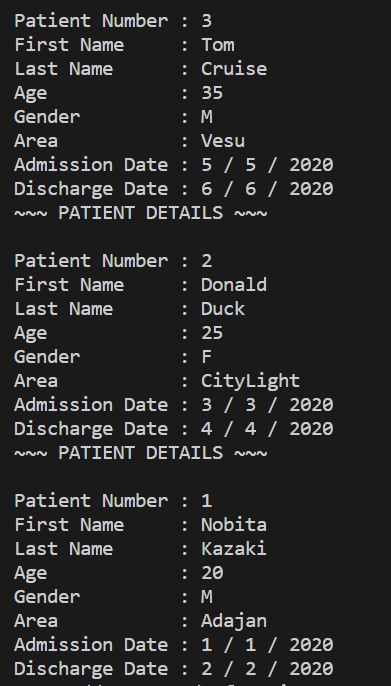
**D) Sorting in Asecnding Order Based on Age**





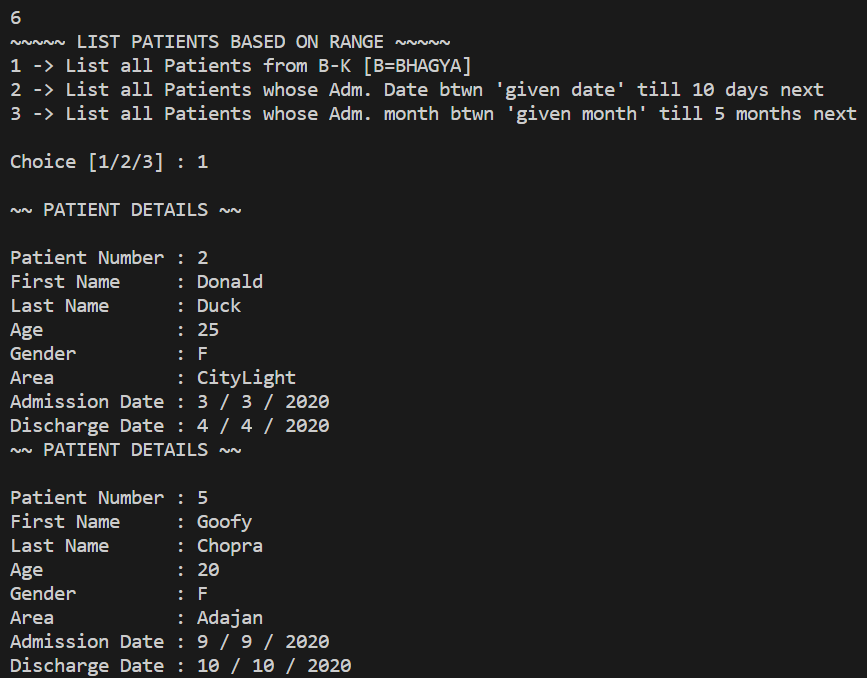
**E) Sorting in Descending Order Based on Discharge Date**



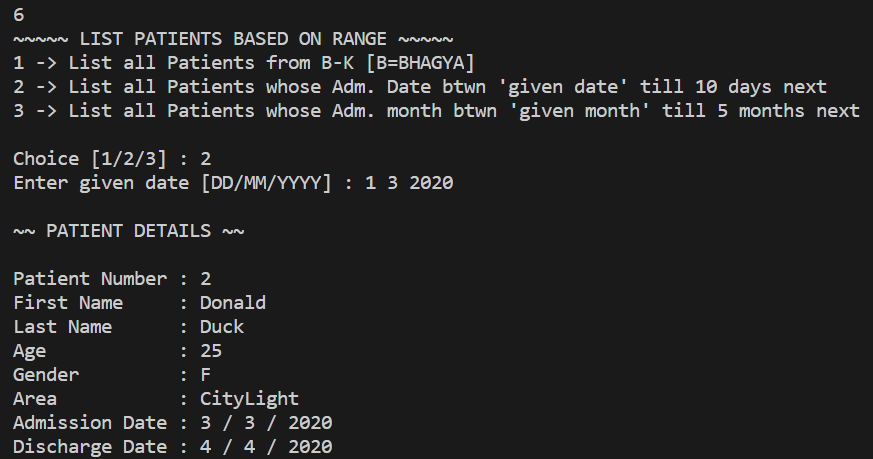


**4) Specific Range Records**

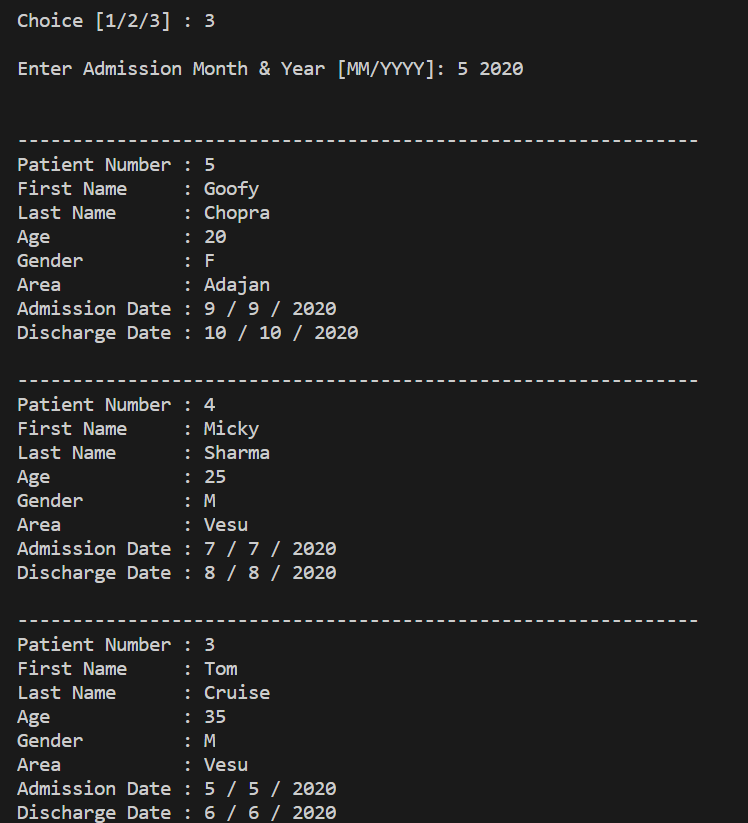
**A) Patients from Letter B to K**



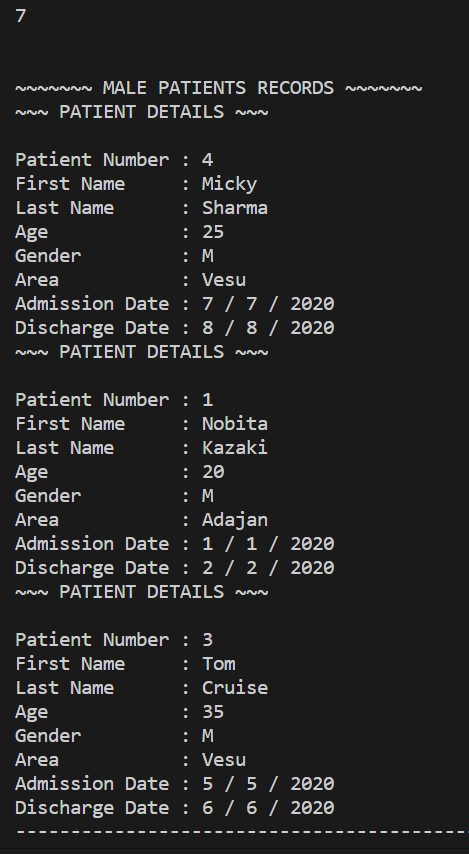
**B) List Patients from 1/3/2020**

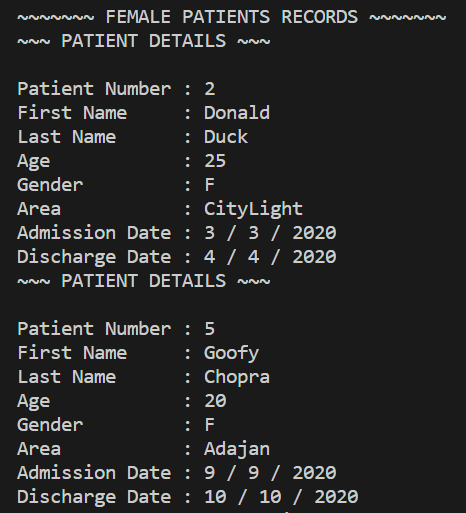


**C) List all Patients from 5/2020 to 10/2020**

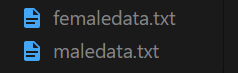


**5) Separate Records based on Gender**





**Two Files are alo Created!**



**Submitted By:**

**BHAGYA VINOD RANA**

**U19CS012**