```c
/*Description: Create a menu-driven program to manage patient information,
including basic details, medical history,
and current medications.
Menu Options:
Add New Patient
View Patient Details
Update Patient Information
Delete Patient Record
List All Patients
Exit
Requirements:
Use variables to store patient details.
Utilize static and const for immutable data such as hospital name.
Implement switch case for menu selection.
Employ loops for iterative tasks like listing patients.
Use pointers for dynamic memory allocation.
Implement functions for CRUD operations.
Utilize arrays for storing multiple patient records.
Use structures for organizing patient data.
Apply nested structures for detailed medical history.
Use unions for optional data fields.
Employ nested unions for multi-type data entries.*/
#include<stdio.h>
#include<stdlib.h>
#include<string.h>
#define max 20
struct Medicalhistory
{
    float BP;
    float sugar;
    float cholestrol;
};
struct Personal
{
    char name[20];
    int id;
    int age;
    float weight;
    float height;
};
```

```c
union Class
{
    char vip[10];
    char norm[10];
};
struct Medication
{
    char m1[10];
    char m2[10];
    char m3[10];
};
struct Patient
{
    struct Personal personal;
    struct Medicalhistory mhistory;
    struct Medication medication;
    union Class class;

};
static int total_patients=0;
struct Patient patient[20];
void add();
void view();
void update();
void delete();
void list();
void main()
{int choice;
    do
    {
        printf("1.Add new patient\n");
        printf("2.View patient details\n");
        printf("3.Update patient informaton\n");
        printf("4.Delete patient record\n");
        printf("5.List all patientt\n");
        printf("6.Exit\n");
        printf("Enter choice\n");
        scanf("%d",&choice);
        switch(choice)
        {
```

```
            case 1:
            add();
            break;
            case 2:
            view();
            break;
            case 3:
            update();
            break;
            case 4:
            delete();
            break;
            case 5:list();
            break;
            case 6:printf("Exiting....\n");
            break;
            default : printf("Enter valid input\n");
            break;
        }
    } while (6!=choice);

}
void add()
{
    if(total_patients>=max)
    {
        printf("No more patients can be admited\n");
        return;
    }
    struct Patient *newpatient=(struct Patient *)malloc(sizeof(struct
Patient));
    newpatient =&patient[total_patients];
    printf("Enter namee of patient : ");
    scanf("%s",newpatient->personal.name);
    printf("Enter age : ");
    scanf("%d",&newpatient->personal.age);
    printf("Enter height : ");
    scanf("%f",&newpatient->personal.height);
    printf("Enter weight : ");
    scanf("%f",&newpatient->personal.weight);
```

```
    printf("Enter BP : ");
    scanf("%f",&newpatient->mhistory.BP);
    printf("Enter sugar lvl : ");
    scanf("%f",&newpatient->mhistory.sugar);
    printf("Enter cholestrol : ");
    scanf("%f",&newpatient->mhistory.cholestrol);
    printf("Enter medication 1 :");
    scanf("%s",newpatient->medication.m1);
       printf("Enter medication 2 :");
    scanf("%s",newpatient->medication.m2);
       printf("Enter medication 3 :");
    scanf("%s",newpatient->medication.m3);
    newpatient->personal.id=total_patients+1;
    total_patients++;
    printf("patient added to system\n");
}
void view()
{
    int id;
    printf("Enter patient id : ");
    scanf("%d",&id);
    if(id!=0)
    {
           for(int i=0;i<total_patients;i++)
    {
            if(id==patient[i].personal.id)
        {
            printf("Name : %s\n",patient[i].personal.name);
            printf("Age : %d\n",patient[i].personal.age);
            printf("Height : %.2f\n",patient[i].personal.height);
            printf("Weight : %.2f\n",patient[i].personal.weight);
            printf("BP : %.2f\n",patient[i].mhistory.BP);
             printf("Cholestrol : %.2f\n",patient[i].mhistory.cholestrol);
              printf("Sugar levels : %.2f\n",patient[i].mhistory.sugar);
              printf("Medications : %s |%s
|%s\n",patient[i].medication.m1,patient[i].medication.m2,patient[i].medica
tion.m3);
    }
    }
    }
```

```
    else
    printf("Record not found\n");

}

void update()
{
    int id;
    printf("Enter patient id : ");
    scanf("%d",&id);
    for(int i=0;i<total_patients;i++)
    {
        if(id!=0)
        {
                if(id==patient[i].personal.id)
        {
             printf("Enter height : ");
        scanf("%f",&patient[i].personal.height);
        printf("Enter weight : ");
        scanf("%f",&patient[i].personal.weight);
        printf("Enter BP : ");
        scanf("%f",&patient[i].mhistory.BP);
        printf("Enter sugar lvl : ");
        scanf("%f",&patient[i].mhistory.sugar);
        printf("Enter cholestrol : ");
        scanf("%f",&patient[i].mhistory.cholestrol);
        printf("Enter medication 1 :");
        scanf("%s",patient[i].medication.m1);
       printf("Enter medication 2 :");
        scanf("%s",patient[i].medication.m2);
       printf("Enter medication 3 :");
        scanf("%s",patient[i].medication.m3);
        }
        }
        else
        printf("No record found\n");
    }
}
void delete()
{
```

```
    int id;
    printf("Enter patient id : ");
    scanf("%d",&id);
    if(id!=0)
    {
        for(int i=0;i<total_patients;i++)
        {
            if(id==patient[i].personal.id)
            {
                patient[i].personal.id=0;
                total_patients--;
            }
        }
    }
    else
    printf("There is no record\n");
}
void list()
{
    if(total_patients>0)
    {
         for (int i=0;i<total_patients;i++)
    {
        if(patient[i].personal.id!=0)
        {
            printf("Patient id : %d\n",patient[i].personal.id);
        printf("Name : %s\n",patient[i].personal.name);
            printf("Age : %d\n",patient[i].personal.age);
            printf("Height : %.2f\n",patient[i].personal.height);
            printf("Weight : %.2f\n",patient[i].personal.weight);
            printf("BP : %.2f\n",patient[i].mhistory.BP);
             printf("Cholestrol : %.2f\n",patient[i].mhistory.cholestrol);
              printf("Sugar levels : %.2f\n",patient[i].mhistory.sugar);
              printf("Medications : %s |%s
|%s\n",patient[i].medication.m1,patient[i].medication.m2,patient[i].medica
tion.m3);

        }
```

```
    }
    }
    else
    printf("There are no patients\n");

}
```

```
PS D:\projects\quest\c> cd "d:\projects\ques
1.Add new patient
2.View patient details
3.Update patient informaton
4.Delete patient record
5.List all patientt
6.Exit
Enter choice
1
Enter namee of patient : red
Enter age : 15
Enter height : 121
Enter weight : 21
Enter BP : 78
Enter sugar lvl : 100
Enter cholestrol : 13
Enter medication 1 :avas5
Enter medication 2 :morpine
Enter medication 3 :stim
patient added to system
1.Add new patient
2.View patient details
3.Update patient informaton
4.Delete patient record
5.List all patientt
6.Exit
Enter choice
2
Enter patient id : 1
Name : red
Age : 15
Height : 121.00
Weight : 21.00
BP : 78.00
Cholestrol : 13.00
Sugar levels : 100.00
Medications : avas5 |morpine |stim
1.Add new patient
2.View patient details
3.Update patient informaton
4.Delete patient record
```

```c
#include<stdio.h>
#include<string.h>
#include<stdlib.h>
#define max 20
union Unit
{
    int litre;
    int piece;
    int kilo;
};
struct Price
{
    float p;

};
struct Item
{
    int id;
    char name[10];
    struct Price price;
    char unit_type[6];
    union Unit unit;


};
struct Item item[20];
static int totalitems=0;
void add();
void view();
void update();
void delete();
void list();
void main()
{
    int choice;
    do
    {
        printf("1.Add inventory item\n");
        printf("2.View inventory item\n");
        printf("3.Update inventory item\n");
```

```
        printf("4.Delete inventory item\n");
        printf("5.List all inventory iitem\n");
        printf("6.Exit\n");
        printf("Enter choice\n");
        scanf("%d",&choice);
        switch(choice)
        {
            case 1:
            add();
            break;
            case 2:
            view();
            break;
            case 3:
            update();
            break;
            case 4:
            delete();
            break;
            case 5:list();
            break;
            case 6:printf("Exiting....\n");
            break;
            default : printf("Enter valid input\n");
            break;
        }
    } while (6!=choice);

}
void add()
{
    if(totalitems>=max)
    {
     printf("All slots are filled\n");
     return;
    }
    struct Item * newitem =(struct Item *)malloc(sizeof(struct Item ));
    newitem =&item[totalitems];
    printf("Enter item name : ");
    scanf("%s",newitem->name);
```

```
    printf("Enter item price : ");
    scanf("%f",&newitem->price.p);
    printf("Enter unit type : ");
    scanf("%s",newitem->unit_type);
    if(strcmp(newitem->unit_type,"kilo")==0)
    {
        printf("Enter total kilos of item : ");
        scanf("%d",&newitem->unit.kilo);
    }
    else if(strcmp(newitem->unit_type,"litre")==0)
    {
        printf("Enter total litres of item : ");
        scanf("%d",&newitem->unit.litre);
    }
    else
    {
        printf("Enter total pieces of item : ");
        scanf("%d",&newitem->unit.piece);
    }
    newitem->id=totalitems+1;
    totalitems++;
    printf("Item added to inventory\n");

}
void view()
{
    int id;
    printf("Enter itemid : ");
    scanf("%d",&id);
    if(id!=0)
    {
           for(int i=0;i<totalitems;i++)
    {
            if(id==item[i].id)
        {
            printf("Name : %s\n",item[i].name);
            printf("Price : %.2f\n",item[i].price.p);
            if(strcmp(item[i].unit_type,"kilo")==0)
            printf("Total weight of item : %d\n",item[i].unit.kilo);
            else if(strcmp(item[i].unit_type,"litre")==0)
```

```
            printf("Total volume of item : %d\n",item[i].unit.litre);
            else if(strcmp(item[i].unit_type,"piece")==0)
            printf("Total number of pieces of item :
%d\n",item[i].unit.piece);
            else;

    }
    }
    }
    else
    printf("Record not found\n");

}
void update()
{
    int id;
    printf("Enter item id : ");
    scanf("%d",&id);
    for(int i=0;i<totalitems;i++)
    {
        if(id!=0)
        {
                if(id==item[i].id)
        {
             printf("Enter item price : ");
    scanf("%f",&item[i].price.p);
    printf("Enter unit type : ");
    scanf("%s",item[i].unit_type);
    if(strcmp(item[i].unit_type,"kilo")==0)
    {
        printf("Enter total kilos of item : ");
        scanf("%d",&item[i].unit.kilo);
    }
    else if(strcmp(item[i].unit_type,"litre")==0)
    {
        printf("Enter total litres of item : ");
        scanf("%d",&item[i].unit.litre);
    }
    else
    {
```

```
        printf("Enter total pieces of item : ");
        scanf("%d",&item[i].unit.piece);
    }
        }
        }
        else
        printf("No item found\n");
    }
}
void delete()
{
    int id;
    printf("Enter item id : ");
    scanf("%d",&id);
    if(id!=0)
    {
        for(int i=0;i<totalitems;i++)
        {
            if(id==item[i].id)
            {
                item[i].id=0;
                totalitems--;
            }
        }
    }
    else
    printf("There is no item\n");
}
void list()
{
    if(totalitems>0)
    {
         for (int i=0;i<totalitems;i++)
    {
        if(item[i].id!=0)
        {
            printf("Name : %s\n",item[i].name);
            printf("Price : %.2f\n",item[i].price.p);
            if(strcmp(item[i].unit_type,"kilo")==0)
            printf("Total weight of item : %d\n",item[i].unit.kilo);
```

```
            else if(strcmp(item[i].unit_type,"litre")==0)
            printf("Total volume of item : %d\n",item[i].unit.litre);
            else if(strcmp(item[i].unit_type,"piece")==0)
            printf("Total number of pieces of item :
%d\n",item[i].unit.piece);
            else;

        }
        printf("\n");
    }
    }
    else
    printf("There are no items\n");

}
```

```
Enter item name : milk
Enter item price : 15
Enter unit type : litre
Enter total litres of item : 100
Item added to inventory
1.Add inventory item
2.View inventory item
3.Update inventory item
4.Delete inventory item
5.List all inventory iitem
6.Exit
Enter choice
2
Enter itemid : 2
Name : milk
Price : 15.00
Total volume of item : 100
1.Add inventory item
2.View inventory item
3.Update inventory item
4.Delete inventory item
5.List all inventory iitem
6.Exit
Enter choice
5
Name : pen
Price : 10.00
Total number of pieces of item : 100

Name : milk
Price : 15.00
Total volume of item : 100

1.Add inventory item
2.View inventory item
3.Update inventory item
4.Delete inventory item
5.List all inventory iitem
6.Exit
Enter choice
```

```c
/*Description: Develop a system to manage patient appointments.
Menu Options:
Schedule Appointment
View Appointment
Update Appointment
Cancel Appointment
List All Appointments
Exit
Requirements:
Use variables for appointment details.
Apply static and const for non-changing data like clinic hours.
Implement switch case for appointment operations.
Utilize loops for scheduling.
Use pointers for dynamic data manipulation.
Create functions for appointment handling.
Use arrays for storing appointments.
Define structures for appointment details.
Employ nested structures for detailed doctor and patient information.
Utilize unions for optional appointment data.
Apply nested unions for complex appointment data.*/
#include<stdio.h>
#include<string.h>
#include<stdlib.h>
#define max 20
struct Doctor
{
    char dname[20];
    char did[5];

};
struct Patient
{
    char name[20];
    char pid[5];
};
union Specialization
{
    char ortho[15];
    char gastric[15];
    char pulmo[15];
```

```
};
union Type
{
    int first;
    int follow;
};
struct Appointment
{
    int apno;
    struct Doctor doctor;
    struct Patient patient;
    char stype;
    union Specialization specialization;
    char visit[10];
    union Type type;

};
static int total=0;
struct Appointment appointment[20];

void add();
void view();
void update();
void delete();
void list();
void main()
{
    int choice;
    do
    {
        printf("1.Schedule appoitment\n");
        printf("2.View appointement\n");
        printf("3.Update appointement\n");
        printf("4.Cancel appointment\n");
        printf("5.List all appointment\n");
        printf("6.Exit\n");
        printf("Enter choice\n");
        scanf("%d",&choice);
        switch(choice)
```

```
        {
            case 1:
            add();
            break;
            case 2:
            view();
            break;
            case 3:
            update();
            break;
            case 4:
            delete();
            break;
            case 5:list();
            break;
            case 6:printf("Exiting....\n");
            break;
            default : printf("Enter valid input\n");
            break;
        }
    } while (6!=choice);

}
void add()
{
    if(total>=max)
    {
     printf("All slots are filled\n");
     return;
    }
    struct Appointment * newapp =(struct Appointment
*)malloc(sizeof(struct Appointment ));
    newapp =&appointment[total];
    printf("Enter doctor id : ");
    scanf("%s",newapp->doctor.did);
    printf("Enter doctor name : ");
    scanf("%s",newapp->doctor.dname);
    printf("Enter patient id : ");
    scanf("%s",newapp->patient.pid);
    printf("Enter patient id : ");
```

```
    scanf("%s",newapp->patient.name);
    printf("Enter specialiation (g.gastro,o,ortho,p.pulmonology) :");
    getchar();
    scanf("%c",&newapp->stype);
    if(newapp->stype=='g')
    strcpy(newapp->specialization.gastric,"Gastric");
    else if(newapp->stype=='o')
    strcpy(newapp->specialization.ortho,"Ortho");
    else if(newapp->stype=='p')
    strcpy(newapp->specialization.pulmo,"Pulomonology");
    newapp->apno=total+1;
    printf("Enter visit type :");
    scanf("%s",newapp->visit);
    if(strcmp(newapp->visit,"first")==0)
    newapp->type.first=1;
    else if(strcmp(newapp->visit,"followup")==0)
    {
        printf("Enter followup num : ");
        scanf("%d",&newapp->type.follow);
    }
    total++;
    printf("Item added to inventory\n");

}
void view()
{
    int no;
    printf("Enter appointment no : ");
    scanf("%d",&no);
    if(no!=0)
    {
           for(int i=0;i<total;i++)
    {
            if(no==appointment[i].apno)
        {
            printf("Patient id : %s\n",appointment[i].patient.pid);
            printf("Patient name : %s\n",appointment[i].patient.name);
            printf("Doctor id : %s\n",appointment[i].doctor.did);
            printf("Doctor name : %s\n",appointment[i].doctor.dname);
            if(appointment[i].stype=='g')
```

```
            printf("Specialization :
%s\n",appointment[i].specialization.gastric);
            else if(appointment[i].stype=='o')
            printf("Specialization :
%s\n",appointment[i].specialization.ortho);
            else if(appointment[i].stype=='p')
            printf("Specialization :
%s\n",appointment[i].specialization.pulmo);
            if(strcmp(appointment[i].visit,"first")==0)
            printf("First visit\n");
            else
            printf("Folowup : %d\n",appointment[i].type.follow);
        }
    }
    }
    else
    printf("Record not found\n");

}
void list()
{
    if(total>0)
    {
         for (int i=0;i<total;i++)
    {
        if(appointment[i].apno!=0)
        {
            printf("Patient id : %s\n",appointment[i].patient.pid);
            printf("Patient name : %s\n",appointment[i].patient.name);
            printf("Doctor id : %s\n",appointment[i].doctor.did);
            printf("Doctor name : %s\n",appointment[i].doctor.dname);
            if(appointment[i].stype=='g')
            printf("Specialization :
%s\n",appointment[i].specialization.gastric);
            else if(appointment[i].stype=='o')
            printf("Specialization :
%s\n",appointment[i].specialization.ortho);
            else if(appointment[i].stype=='p')
            printf("Specialization :
%s\n",appointment[i].specialization.pulmo);
```

```
            if(strcmp(appointment[i].visit,"first")==0)
            printf("First visit\n");
            else
            printf("Folowup : %d\n",appointment[i].type.follow);

printf("\n");
        }
        printf("\n");
    }
    }
    else
    printf("There are no appointments\n");

}
void update()
{
    int no;
    printf("Enter appointment no : ");
    scanf("%d",&no);
    for(int i=0;i<total;i++)
    {
        if(no!=0)
        {
                if(no==appointment[i].apno)
        {
     printf("Enter octor id :");
     scanf("%s",appointment[i].doctor.did);
    printf("Enter doctor name : ");
    scanf("%s",appointment[i].doctor.dname);
      printf("Enter specialiation (g.gastro,o,ortho,p.pulmonology) :");
    getchar();
    scanf("%c",&appointment[i].stype);
    if(appointment[i].stype=='g')
    strcpy(appointment[i].specialization.gastric,"Gastric");
    else if(appointment[i].stype=='o')
    strcpy(appointment[i].specialization.ortho,"Ortho");
    else if(appointment[i].stype=='p')
    strcpy(appointment[i].specialization.pulmo,"Pulomonology");
 printf("Enter visit type :");
    scanf("%s",appointment[i].visit);
```

```
    if(strcmp(appointment[i].visit,"first")==0)
    appointment[i].type.first=1;
    else if(strcmp(appointment[i].visit,"followup")==0)
    {
        printf("Enter followup num : ");
        scanf("%d",&appointment[i].type.follow);
    }
printf("Appointment updated\n");
        }
        }
        else
        printf("No item found\n");
    }
}

void delete()
{
    int no;
    printf("Enter appoint no : ");
    scanf("%d",&no);
    if(no!=0)
    {
        for(int i=0;i<total;i++)
        {
            if(no==appointment[i].apno)
            {
                appointment[i].apno=0;
                total--;
            }
        }
    }
    else
    printf("There is no appointment\n");
}
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

```
Enter choice
3
Enter appointment no : 2
Enter octor id :doc1
Enter doctor name : doc
Enter specialiation (g.gastro,o,ortho,p.pulmonology)
Enter visit type :followup
Enter followup num : 5
Appointment updated
1.Schedule appoitment
2.View appointement
3.Update appointement
4.Cancel appointment
5.List all appointment
6.Exit
Enter choice
5
Patient id : p1
Patient name : pat1
Doctor id : d1
Doctor name : doc1
Specialization : Gastric
First visit


Patient id : p1
Patient name : pat1
Doctor id : doc1
Doctor name : doc
Specialization : Ortho
Folowup : 5


1.Schedule appoitment
2.View appointement
3.Update appointement
4.Cancel appointment
5.List all appointment
6.Exit
Enter choice
```

```c
/*Description: Create a billing system for patients.
Menu Options:
Generate Bill
View Bill
Update Bill
Delete Bill
List All Bills
Exit
Requirements:
Declare variables for billing information.
Use static and const for fixed billing rates.
Implement switch case for billing operations.
Utilize loops for generating bills.
Use pointers for bill calculations.
Create functions for billing processes.
Use arrays for storing billing records.
Define structures for billing components.
Employ nested structures for detailed billing breakdown.
Use unions for variable billing elements.
Apply nested unions for complex billing scenarios.*/
#include<stdio.h>
#include<string.h>
#include<stdlib.h>
#define max 20
static int totalbills=0;
struct Item
{
    int itemid;
    float price;
};
union Discount
{
    float special;
    float member;
};
struct Bills
{
    int  biilno;
    int items;
    struct Item item[5];
```

```
    char type;
    union Discount discount;
    float total;
};
struct Bills bill[20];
void add();
void view();
void update();
void delete();
void list();
void main()
{
    int choice;
    do
    {
        printf("1.Generate bill\n");
        printf("2.View bill\n");
        printf("3.Update bill\n");
        printf("4.Delete bill\n");
        printf("5.List all bills\n");
        printf("6.Exit\n");
        printf("Enter choice\n");
        scanf("%d",&choice);
        switch(choice)
        {
            case 1:
            add();
            break;
            case 2:
            view();
            break;
            case 3:
            update();
            break;
            case 4:
            delete();
            break;
            case 5:list();
            break;
            case 6:printf("Exiting....\n");
```

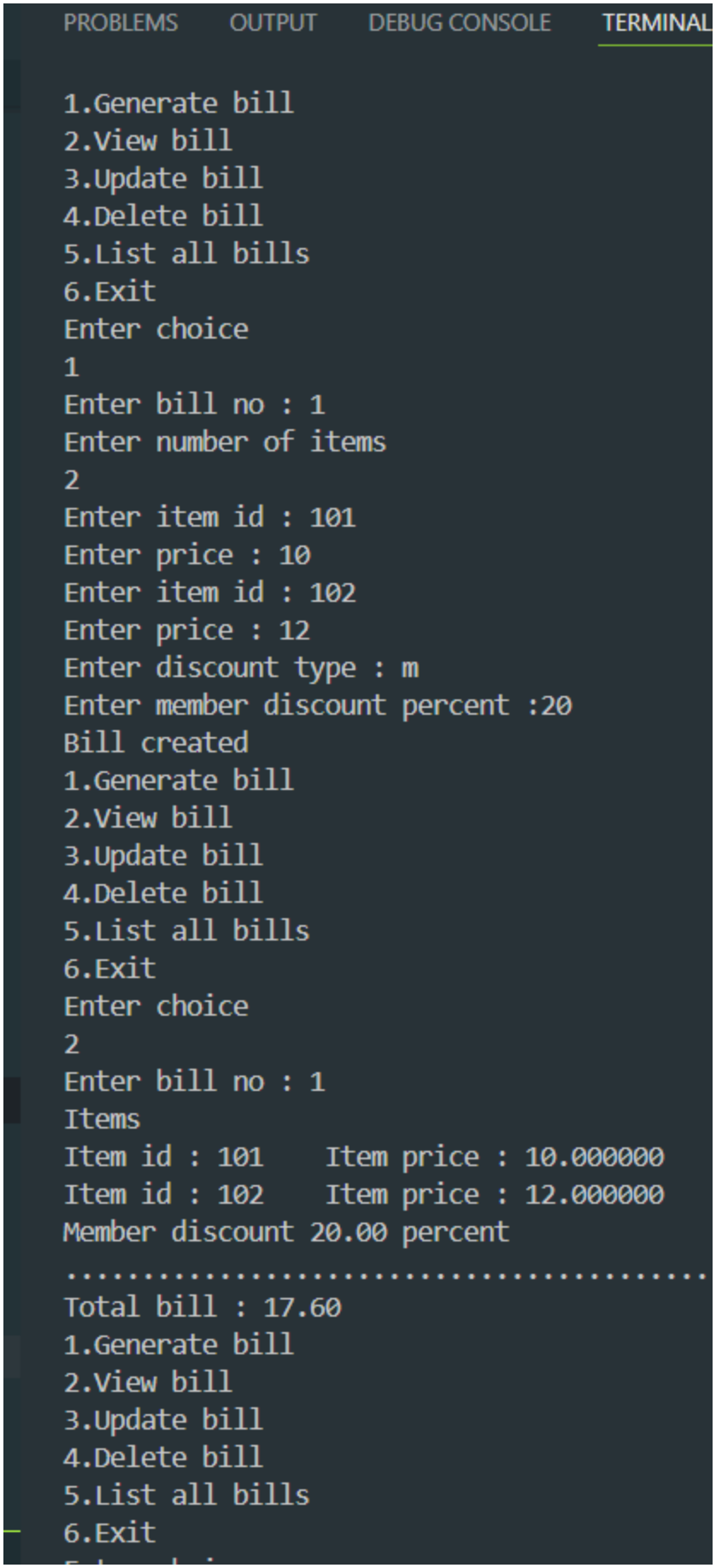
```
            break;
            default : printf("Enter valid input\n");
            break;
        }
    } while (6!=choice);

}
void add()
{
    int sum=0;
    if(totalbills>=max)
    {
     printf("All slots are filled\n");
     return;
    }
    struct Bills * newbill =(struct Bills *)malloc(sizeof(struct Bills ));
    newbill =&bill[totalbills];
    printf("Enter bill no : ");
    scanf("%d",&newbill->biilno);
    printf("Enter number of items\n");
    scanf("%d",&newbill->items);
    for(int i=0;i<newbill->items;i++)
    {
        printf("Enter item id : ");
        scanf("%d",&newbill->item[i].itemid);
        printf("Enter price : ");
        scanf("%f",&newbill->item[i].price);
        sum+=newbill->item[i].price;
    }
    printf("Enter discount type : ");
    getchar();
    scanf("%c",&newbill->type);
    if(newbill->type=='m')
    {
        printf("Enter member discount percent :");
        scanf("%f",&newbill->discount.member);
        newbill->total=sum -(sum*(newbill->discount.member))/100;
    }
    else
    {
```

```
        printf("Enter vip discount percent : ");
        scanf("%f",&newbill->discount.special);
        newbill->total=sum-(sum*(newbill->discount.special))/100;
    }

    totalbills++;
    printf("Bill created\n");

}
void view()
{
    int no;
    printf("Enter bill no : ");
    scanf("%d",&no);
    if(no!=0)
    {
           for(int i=0;i<totalbills;i++)
    {
            if(no==bill[i].biilno)
        {
            printf("Items\n");
            for(int j=0;j<bill[i].items;j++)
            {
                printf("Item id : %d\t Item price :
%f\n",bill[i].item[j].itemid,bill[i].item[j].price);
            }
            if(bill[i].type=='m')
            printf("Member discount %.2f
percent\n",bill[i].discount.member);
            else
            printf("VIP discount %.2f
percent\n",bill[i].discount.special);
            printf(".....\n");
            printf("Total bill : %.2f\n",bill[i].total);
        }
    }
    }
    else
    printf("Bill not found\n");
```

```
}
void list()
{
    if(totalbills>0)
    {
         for (int i=0;i<totalbills;i++)
    {
           if(bill[i].biilno!=0)
        {
            printf("Items\n");
            for(int j=0;j<bill[i].items;j++)
            {
                printf("Item id : %d\t Item price :
%f\n",bill[i].item[j].itemid,bill[i].item[j].price);
            }
            if(bill[i].type=='m')
            printf("Member discount %.2f
percent\n",bill[i].discount.member);
            else
            printf("VIP discount %.2f
percent\n",bill[i].discount.special);
            printf(".....\n");
            printf("Total bill : %.2f\n",bill[i].total);
        }
        printf("\n");
    }
    }
    else
    printf("There are no bills\n");

}
void update()
{
    int no,sum=0;
    printf("Enter bill no : ");
    scanf("%d",&no);
    if(no!=0)
    {
         for(int i=0;i<totalbills;i++)
    {
```

```
        if(no==bill[i].biilno)
        {
             printf("Enter number of items\n");
    scanf("%d",&bill[i].items);
    for(int j=0;i<bill[i].items;j++)
    {
        printf("Enter item id : ");
        scanf("%d",&bill[i].item[j].itemid);
        printf("Enter price : ");
        scanf("%f",&bill[i].item[j].price);
        sum+=bill[i].item[j].price;
    }
    printf("Enter discount type : ");
    scanf("%c",&bill[i].type);
    if(bill[i].type=='m')
    {
        printf("Enter member discount percent :");
        scanf("%f",&bill[i].discount.member);
        bill[i].total=sum*(100-(bill[i].discount.member));
    }
    else
    {
        printf("Enter vip discount percent : ");
        scanf("%f",&bill[i].discount.special);
        bill[i].total=sum*(100-(bill[i].discount.special));
    }

        }
    }

    }
    else
    printf("No bill found\n");

}

void delete()
{
    int no;
    printf("Enter bill no : ");
```

```
    scanf("%d",&no);
    if(no!=0)
    {
        for(int i=0;i<totalbills;i++)
        {
            if(no==bill[i].biilno)
            {
                bill[i].biilno=0;
                totalbills--;
            }
        }
    }
    else
    printf("There is no bill\n");
}
```


PROBLEMS
OUTPUT
DEBUG CONSOLE
TERMINAL
1.Generate bill
2.View bill
3.Update bill
4.Delete bill
5.List all bills
6.Exit
Enter choice
1
Enter bill no : 1
Enter number of items
2
Enter item id : 101
Enter price : 10
Enter item id : 102
Enter price : 12
Enter discount type : m
Enter member discount percent :20
Bill created
1.Generate bill
2.View bill
3.Update bill
4.Delete bill
5.List all bills
6.Exit
Enter choice
2
Enter bill no : 1
Items
Item id : 101 Item price : 10.000000
Item id : 102 Item price : 12.000000
Member discount 20.00 percent
.....
Total bill : 17.60
1.Generate bill
2.View bill
3.Update bill
4.Delete bill
5.List all bills
6.Exit

```c
/*Description: Develop a system to manage and store patient test results
Menu Options:
Add Test Result
View Test Result
Update Test Result
Delete Test Result
List All Test Results
Exit
Requirements:
Declare variables for test results.
Use static and const for standard test ranges.
Implement switch case for result operations.
Utilize loops for result input and output.
Use pointers for handling result data.
Create functions for result management.
Use arrays for storing test results.
Define structures for test result details.
Employ nested structures for detailed test parameters.
Utilize unions for optional test data.
Apply nested unions for complex test result data*/
#include<stdio.h>
#include<string.h>
#include<stdlib.h>
#define max 20
static int totaltest=0;
struct Type
{
    float BP;
    float sugar;
    float cholestrol;

};
union Bt
{
    char positive[2];
    char negative[2];
};
struct Test
{
    int id;
```

```
    struct Type type;
    char btype;
    union Bt bt;


};
struct Test test[20];
void add();
void view();
void update();
void delete();
void list();
void main()
{
    int choice;
    do
    {
        printf("1.Add test result\n");
        printf("2.View test result\n");
        printf("3.Update test result\n");
        printf("4.Delete test resut\n");
        printf("5.List all test result\n");
        printf("6.Exit\n");
        printf("Enter choice\n");
        scanf("%d",&choice);
        switch(choice)
        {
            case 1:
            add();
            break;
            case 2:
            view();
            break;
            case 3:
            update();
            break;
            case 4:
            delete();
            break;
            case 5:list();
```

```
            break;
            case 6:printf("Exiting....\n");
            break;
            default : printf("Enter valid input\n");
            break;
        }
    } while (6!=choice);

}
void add()
{
    if(totaltest>=max)
    {
     printf("All slots are filled\n");
     return;
    }
    struct Test * newtest =(struct Test *)malloc(sizeof(struct Test ));
    newtest =&test[totaltest];
    printf("Enter test id: ");
    scanf("%d",&newtest->id);
    printf("Enter sugar lvl : \n");
    scanf("%f",&newtest->type.sugar);
     printf("Enter BP lvl : \n");
    scanf("%f",&newtest->type.BP);
     printf("Enter Cholestrol lvl : \n");
    scanf("%f",&newtest->type.cholestrol);
    printf("Enter blood rh factor : ");
    getchar();
    scanf("%c",&newtest->btype);
    printf("Enter blood type : ");
    if(newtest->btype='+')
    scanf("%s",newtest->bt.positive);
    else
    scanf("%s",newtest->bt.negative);
    totaltest++;
    printf("test created\n");

}
void view()
{
```

```
    int id;
    printf("Enter test id : ");
    scanf("%d",&id);
    if(id!=0)
    {
            for(int i=0;i<totaltest;i++)
    {
             if(id==test[i].id)
        {
             printf("Sugar level : %.2f\n",test[i].type.sugar);
             printf("BP level : %.2f\n",test[i].type.BP);
             printf("Cholestrol level : %.2f\n",test[i].type.cholestrol);
             if(test[i].btype='+')
             printf("Blood type is
%s%cve\n",test[i].bt.positive,test[i].btype);
             else
             printf("Blood type is
%s%cve\n",test[i].bt.negative,test[i].btype);
        }
    }
    }
    else
    printf("Test result not found\n");

}
void list()
{
    if(totaltest>0)
    {
         for (int i=0;i<totaltest;i++)
    {
           if(test[i].id!=0)
        {   printf("Test id : %d\n",test[i].id);
             printf("Sugar level : %.2f\n",test[i].type.sugar);
             printf("BP level : %.2f\n",test[i].type.BP);
             printf("Cholestrol level : %.2f\n",test[i].type.cholestrol);
             if(test[i].btype='+')
             printf("Blood type is
%s%cve\n",test[i].bt.positive,test[i].btype);
             else
```

```
            printf("Blood type is %s
%cve\n",test[i].bt.negative,test[i].btype);
        }
        printf("\n");
    }
    }
    else
    printf("There are no test results\n");

}
void update()
{
    int id;
    printf("Enter test id : ");
    scanf("%d",&id);
    if(id!=0)
    {
         for(int i=0;i<totaltest;i++)
    {
        if(id==test[i].id)
        {
              printf("Enter sugar lvl : \n");
    scanf("%f",&test[i].type.sugar);
     printf("Enter BP lvl : \n");
    scanf("%f",&test[i].type.BP);
     printf("Enter Cholestrol lvl : \n");
    scanf("%f",&test[i].type.cholestrol);
    printf("Enter blood rh factor : ");
    scanf("%c",&test[i].btype);
    printf("Enter blood type : ");
    if(test[i].btype='+')
    scanf("%s",test[i].bt.positive);
    else
    scanf("%s",test[i].bt.negative);
        }
    }

    }
    else
    printf("No test result found\n");
```

```
}

void delete()
{
    int id;
    printf("Enter test id : ");
    scanf("%d",&id);
    if(id!=0)
    {
        for(int i=0;i<totaltest;i++)
        {
            if(id==test[i].id)
            {
                test[i].id=0;
                totaltest--;
            }
        }
    }
    else
    printf("There is no test result\n");
}
```

```
Enter blood type : AB
test created
1.Add test result
2.View test result
3.Update test result
4.Delete test resut
5.List all test result
6.Exit
Enter choice
2
Enter test id : 1
Sugar level : 200.00
BP level : 100.00
Cholestrol level : 15.00
Blood type is AB+ve
1.Add test result
2.View test result
3.Update test result
4.Delete test resut
5.List all test result
6.Exit
Enter choice
4
Enter test id : 1
1.Add test result
2.View test result
3.Update test result
4.Delete test resut
5.List all test result
6.Exit
Enter choice
5
There are no test results
1.Add test result
2.View test result
3.Update test result
4.Delete test resut
5.List all test result
6.Exit
Enter choice
```

```c
/*Description: Implement a linked list to manage a queue of patients
waiting for consultation. Operations:
Create a new patient queue.
Insert a patient into the queue.
Display the current queue of patients*/
#include<stdio.h>
#include<stdlib.h>
#include<string.h>

struct queue
{
    char name[10];
    struct queue *next;
};
struct queue *first=NULL,*ptr;

void create()
{
    int n;
    struct queue *temp;
    printf("Enter queue size : ");
    scanf("%d",&n);
    for(int i=0;i<n;i++)
    {
        struct queue *newnode=(struct queue *)malloc(sizeof(struct
queue));
    newnode->next=NULL;
    strcpy(newnode->name,"\0");
    if(first==NULL)
    {
        first=newnode;
        ptr=newnode;
        temp=newnode;

    }
    else
    {
        temp->next=newnode;
        temp=newnode;
    }
```

```
    }

}
void insert()
{
    if(ptr!=NULL)
    {
        printf("Enter name : ");
        scanf("%s",ptr->name);
        ptr=ptr->next;
    }
    else
    printf("Queue full\n");
}
void display()
{
     struct queue *temp=first;
     while(temp!=NULL)
    {
        if(strcmp(temp->name,"\0")!=0)
        printf("%s ->",temp->name);
        temp=temp->next;

    }
    printf("\n");
}
void main()
{int choice;
do
{
    printf("1.Create queue\n");
    printf("2.Insert\n");
    printf("3.Display\n");
    printf("4.Exit\n");
    printf("Enter choice \n");
    scanf("%d",&choice);
    switch (choice)
    {
    case 1:
        create();
```

```
        break;
        case 2:
        insert();
        break;
        case 3:display();
        break;
        case 4: printf("Exiting ...");
        break;

    default:
        break;
    }

} while (choice !=4);



}
```

```
1.Create queue
2.Insert
3.Display
4.Exit
Enter choice
1
Enter queue size : 4
1.Create queue
2.Insert
3.Display
4.Exit
Enter choice
2
Enter name : red
1.Create queue
2.Insert
3.Display
4.Exit
Enter choice
3
red ->
1.Create queue
2.Insert
3.Display
4.Exit
Enter choice
2
Enter name : blue
1.Create queue
2.Insert
3.Display
4.Exit
Enter choice
3
red ->blue ->
1.Create queue
2.Insert
3.Display
4.Exit
Enter choice
```

```c
/*Description: Use a linked list to allocate beds in a hospital ward.
Operations:
Create a list of available beds.
Insert a patient into an available bed.
Display the current bed allocation.*/
#include<stdio.h>
#include<stdlib.h>
#include<string.h>
struct bed
{
    char name[10];
    int free;//1 not free 0 free
    struct bed*next;
};
struct bed *head=NULL,*ptr;
void create()
{
    int n;
    printf("Enter number of beds : ");
    scanf("%d",&n);
    struct bed *temp;
    for(int i=0;i<n;i++)
    {
        struct bed * node=(struct bed *)malloc(sizeof(struct bed));
        node->free=0;
        node->next=NULL;
        if(head==NULL)
        {
            head=node;
            ptr=node;
            temp=node;

        }
        else
        {
            temp->next=node;
            temp=node;
        }
    }
```

```
}
void allocate()
{
    ptr=head;
    char name[10];
    printf("Enter name: ");
    scanf("%s",name);
    getchar();
    while(ptr!=NULL)
    {
        if(ptr->free==0)
        {
            strcpy(ptr->name,name);
            ptr->free=1;
            return;
        }
        ptr=ptr->next;

    }
    printf("No beds available\n");
}
void display()
{
    ptr=head;
    while(ptr!=NULL)
    {
        if(ptr->free==1)
        {
            printf("Bed allocated to %s \n",ptr->name);
        }
        ptr=ptr->next;
    }
}
void main()
{
    int choice;
    do
    {
        printf("1.Create beds\n");
        printf("2.Allocate bed\n");
```

```
        printf("3.Dislay beds\n");
        printf("4.Exit\n");
        printf("Enter choice : \n");
        scanf("%d",&choice);
        switch (choice)
        {
        case 1:
        create();
            break;
            case 2:
            allocate();
            break;
            case 3:
            display();
            break;
            case 4:
            printf("Exiting ......\n");
            break;

        default:printf("Enter valid choice\n");
            break;
        }
    } while (choice !=4);

}
```

```
2.Allocate bed
3.Dislay beds
4.Exit
Enter choice :
1
Enter number of beds : 5
1.Create beds
2.Allocate bed
3.Dislay beds
4.Exit
Enter choice :
2
Enter name: john
1.Create beds
2.Allocate bed
3.Dislay beds
4.Exit
Enter choice :
3
Bed allocated to john
1.Create beds
2.Allocate bed
3.Dislay beds
4.Exit
Enter choice :
2
Enter name: jerry
1.Create beds
2.Allocate bed
3.Dislay beds
4.Exit
Enter choice :
3
Bed allocated to john
Bed allocated to jerry
1.Create beds
2.Allocate bed
3.Dislay beds
4.Exit
Enter choice :
```

```c
#include<stdio.h>
#include<stdlib.h>
#include<string.h>
struct item
{
    char name[10];
    struct item *next;
};
struct item *head=NULL,*ptr;
void create()
{
    int n;
    printf("Enter size");
    scanf("%d",&n);
    ptr=head;
    for(int i=0;i<n;i++)
    {
        struct item * node=(struct item *)malloc(sizeof(struct item));
        strcpy(node->name,"\0");
        node->next=NULL;
        if(head==NULL)
        {
            head=node;
            ptr=node;
        }
        else
        {
            ptr->next=node;
            ptr=node;
        }
    }
}
void insert()
{
    char name[10];
    printf("Enter item name : ");
    scanf("%s",name);
    ptr=head;
    while(ptr!=0)
    {
```

```
        if(strcmp(ptr->name,"\0")==0)
        {
            strcpy(ptr->name,name);
            return;
        }
        ptr=ptr->next;
    }
}
void display()
{
    ptr=head;
    while(ptr!=NULL)
    {
        if(strcmp(ptr->name,"\0")!=0)
        printf("%s ->",ptr->name);
        ptr=ptr->next;
    }
    printf("\n");
}
void main()
{
    int choice;
    do
    {
        printf("1.Create inventory list\n");
        printf("2.Insert new item\n");
        printf("3.Display inventory item\n");
        printf("4.Exit\n");
        printf("Enter choice : ");
        scanf("%d",&choice);
        switch(choice)
        {
            case 1:
            create();
            break;
            case 2:
            insert();
            break;
            case 3:
            display();
```

```
            break;
            case 4:printf("Exiing..\n");
            break;
        }
    } while (choice!=4);

}
```

```
PS D:\projects\quest\C> cd "d:\
1.Create inventory list
2.Insert new item
3.Display inventory item
4.Exit
Enter choice : 1
Enter size5
1.Create inventory list
2.Insert new item
3.Display inventory item
4.Exit
Enter choice : 2
Enter item name : pen
1.Create inventory list
2.Insert new item
3.Display inventory item
4.Exit
Enter choice : 2
Enter item name : ball
1.Create inventory list
2.Insert new item
3.Display inventory item
4.Exit
Enter choice : 3
pen ->ball ->
1.Create inventory list
2.Insert new item
3.Display inventory item
4.Exit
Enter choice :
```

```
/*Description: Develop a linked list to schedule doctor appointments.
Operations:
Create an appointment list.
Insert a new appointment.
Display all scheduled appointments.*/
```

```c
#include<stdio.h>
#include<stdlib.h>
#include<string.h>
struct appointment
{
    char name[10];
    struct appointment *next;
};
struct appointment *head=NULL,*ptr;
void create()
{
    int n;
    printf("Enter size");
    scanf("%d",&n);
    ptr=head;
    for(int i=0;i<n;i++)
    {
        struct appointment * node=(struct appointment
*)malloc(sizeof(struct appointment));
        strcpy(node->name,"\0");
        node->next=NULL;
        if(head==NULL)
        {
            head=node;
            ptr=node;
        }
        else
        {
            ptr->next=node;
            ptr=node;
        }
    }
}
void insert()
{
    char name[10];
    printf("Enter patient name : ");
    scanf("%s",name);
    ptr=head;
    while(ptr!=0)
```

```
    {
        if(strcmp(ptr->name,"\0")==0)
        {
            strcpy(ptr->name,name);
            return;
        }
        ptr=ptr->next;
    }
}
void display()
{
    ptr=head;
    while(ptr!=NULL)
    {
        if(strcmp(ptr->name,"\0")!=0)
        printf("%s ->",ptr->name);
        ptr=ptr->next;
    }
    printf("\n");
}
void main()
{
    int choice;
    do
    {
        printf("1.Create appointment lst\n");
        printf("2.Insert new patient\n");
        printf("3.Display appointment list\n");
        printf("4.Exit\n");
        printf("Enter choice : ");
        scanf("%d",&choice);
        switch(choice)
        {
            case 1:
            create();
            break;
            case 2:
            insert();
            break;
            case 3:
```

```
            display();
            break;
            case 4:printf("Exiing..\n");
            break;
        }
    } while (choice!=4);

}
```

```
PS D:\projects\quest\C> cd "d:\proje
1.Create appointment lst
2.Insert new patient
3.Display appointment list
4.Exit
Enter choice : 1
Enter size5
1.Create appointment lst
2.Insert new patient
3.Display appointment list
4.Exit
Enter choice : 2
Enter patient name : jerry
1.Create appointment lst
2.Insert new patient
3.Display appointment list
4.Exit
Enter choice : 3
jerry ->
1.Create appointment lst
2.Insert new patient
3.Display appointment list
4.Exit
Enter choice : 2
Enter patient name : tom
1.Create appointment lst
2.Insert new patient
3.Display appointment list
4.Exit
Enter choice : 3
jerry ->tom ->
1.Create appointment lst
2.Insert new patient
3.Display appointment list
4.Exit
Enter choice :
```

```c
/*Description: Implement a linked list to manage emergency contacts for
hospital staff. Operations:
Create a contact list.
Insert a new contact.
Display all emergency contacts.*/
#include<stdio.h>
#include<stdlib.h>
#include<string.h>
struct contact
{
    char name[10];
    char num[10];
    struct contact *next;
};
struct contact *head=NULL,*ptr;
void create()
{
    int n;
    printf("Enter list size : ");
    scanf("%d",&n);
    ptr=head;
    for(int i=0;i<n;i++)
    {
        struct contact * node=(struct contact *)malloc(sizeof(struct
contact));
        strcpy(node->name,"\0");
        strcpy(node->num,"\0");
        node->next=NULL;
        if(head==NULL)
        {
            head=node;
            ptr=node;
        }
        else
        {
            ptr->next=node;
            ptr=node;
        }
    }
}
```

```
void insert()
{
    char name[10],num[10];
    printf("Enter contact name : ");
    scanf("%s",name);
    printf("Enter contact number : ");
    scanf("%s",num);
    ptr=head;
    while(ptr!=0)
    {
        if(strcmp(ptr->name,"\0")==0)
        {
            strcpy(ptr->name,name);
            strcpy(ptr->num,num);
            return;
        }
        ptr=ptr->next;
    }
}
void display()
{
    ptr=head;
    while(ptr!=NULL)
    {
        if(strcmp(ptr->name,"\0")!=0)
        {
            printf("Contact name : %s\n",ptr->name);
       printf("Contact num : %s\n",ptr->num);
        }
        ptr=ptr->next;
    }
    printf("\n");
}
void main()
{
    int choice;
    do
    {
        printf("1.Create contact lst\n");
        printf("2.Insert new contact\n");
```

```
        printf("3.Display contact list\n");
        printf("4.Exit\n");
        printf("Enter choice : ");
        scanf("%d",&choice);
        switch(choice)
        {
            case 1:
            create();
            break;
            case 2:
            insert();
            break;
            case 3:
            display();
            break;
            case 4:printf("Exiing..\n");
            break;
        }
    } while (choice!=4);

}
```

```
1.Create contact lst
2.Insert new contact
3.Display contact list
4.Exit
Enter choice : 2
Enter contact name : ambulance
Enter contact number : 102
1.Create contact lst
2.Insert new contact
3.Display contact list
4.Exit
Enter choice : 3
Contact name : police
Contact num : 100
Contact name : ambulance
Contact num : 102

1.Create contact lst
2.Insert new contact
3.Display contact list
4.Exit
Enter choice : 2
Enter contact name : fire
Enter contact number : 101
1.Create contact lst
2.Insert new contact
3.Display contact list
4.Exit
Enter choice : 3
Contact name : police
Contact num : 100
Contact name : ambulance
Contact num : 102
Contact name : fire
Contact num : 101

1.Create contact lst
2.Insert new contact
3.Display contact list
4.Exit
Enter choice :
```

```c
/*Description: Use a linked list to manage surgery schedules. Operations:
Create a surgery schedule.
Insert a new surgery into the schedule.
Display all scheduled surgeries.*/
#include<stdio.h>
#include<stdlib.h>
#include<string.h>
struct schedule
{
    char name[10];
    struct schedule *next;
};
struct schedule *head=NULL,*ptr;
void create()
{
    int n;
    printf("Enter schedule size : ");
    scanf("%d",&n);
    ptr=head;
    for(int i=0;i<n;i++)
    {
        struct schedule * node=(struct schedule *)malloc(sizeof(struct
schedule));
        strcpy(node->name,"\0");
        node->next=NULL;
        if(head==NULL)
        {
            head=node;
            ptr=node;
        }
        else
        {
            ptr->next=node;
            ptr=node;
        }
    }
}
void insert()
{
    char name[10];
```

```
    printf("Enter patient name : ");
    scanf("%s",name);
    ptr=head;
    while(ptr!=0)
    {
        if(strcmp(ptr->name,"\0")==0)
        {
            strcpy(ptr->name,name);
            return;
        }
        ptr=ptr->next;
    }
}
void display()
{
    ptr=head;
    while(ptr!=NULL)
    {
        if(strcmp(ptr->name,"\0")!=0)
        printf("%s ->",ptr->name);
        ptr=ptr->next;
    }
    printf("\n");
}
void main()
{
    int choice;
    do
    {
        printf("1.Create surgery lst\n");
        printf("2.Insert new patient\n");
        printf("3.Display surgery list\n");
        printf("4.Exit\n");
        printf("Enter choice : ");
        scanf("%d",&choice);
        switch(choice)
        {
            case 1:
            create();
            break;
```

```
            case 2:
            insert();
            break;
            case 3:
            display();
            break;
            case 4:printf("Exiing..\n");
            break;
        }
    } while (choice!=4);

}
```

```
PS D:\projects\quest\C> cd "d:\proje
1.Create surgery lst
2.Insert new patient
3.Display surgery list
4.Exit
Enter choice : 1
Enter schedule size : 4
1.Create surgery lst
2.Insert new patient
3.Display surgery list
4.Exit
Enter choice : 2
Enter patient name : R1
1.Create surgery lst
2.Insert new patient
3.Display surgery list
4.Exit
Enter choice : 2
Enter patient name : R2
1.Create surgery lst
2.Insert new patient
3.Display surgery list
4.Exit
Enter choice : 3
R1 ->R2 ->
1.Create surgery lst
2.Insert new patient
3.Display surgery list
4.Exit
Enter choice :
```

```c
/*Description: Maintain a linked list to keep track of patient history
records. Operations:
Create a history record list.
Insert a new record.
Display all patient history records.*/
#include<stdio.h>
```

```c
#include<stdlib.h>
#include<string.h>
struct schedule
{
    char name[10];
    int age;
    struct schedule *next;
};
struct schedule *head=NULL,*ptr;
void create()
{
    int n;
    printf("Enter schedule size : ");
    scanf("%d",&n);
    ptr=head;
    for(int i=0;i<n;i++)
    {
        struct schedule * node=(struct schedule *)malloc(sizeof(struct
schedule));
        strcpy(node->name,"\0");
        node->age=0;
        node->next=NULL;
        if(head==NULL)
        {
            head=node;
            ptr=node;
        }
        else
        {
            ptr->next=node;
            ptr=node;
        }
    }
}
void insert()
{
    char name[10];
    int  n;
    printf("Enter patient name : ");
    scanf("%s",name);
```

```
    printf("Enter age : ");
    scanf("%d",&n);
    ptr=head;
    while(ptr!=0)
    {
        if(strcmp(ptr->name,"\0")==0)
        {
            strcpy(ptr->name,name);
            ptr->age=n;
            return;
        }
        ptr=ptr->next;
    }
}
void display()
{

    ptr=head;

    while(ptr!=NULL)
    {

        if(strcmp(ptr->name,"\0")!=0)
        {
            printf("Name : %s\t",ptr->name);
            printf("Age : %d",ptr->age);
            printf("\n");
        }
        ptr=ptr->next;
    }
    printf("\n");
}
void main()
{
    int choice;
    do
    {
        printf("1.Create record lst\n");
        printf("2.Insert new record\n");
        printf("3.Display record list\n");
```

```
        printf("4.Exit\n");
        printf("Enter choice : ");
        scanf("%d",&choice);
        switch(choice)
        {
            case 1:
            create();
            break;
            case 2:
            insert();
            break;
            case 3:
            display();
            break;
            case 4:printf("Exiing..\n");
            break;
        }
    } while (choice!=4);

}
```

```
1.Create record lst
2.Insert new record
3.Display record list
4.Exit
Enter choice : 1
Enter schedule size : 5
1.Create record lst
2.Insert new record
3.Display record list
4.Exit
Enter choice : 2
Enter patient name : red
Enter age : 13
1.Create record lst
2.Insert new record
3.Display record list
4.Exit
Enter choice : 2
Enter patient name : blue
Enter age : 11
1.Create record lst
2.Insert new record
3.Display record list
4.Exit
Enter choice : 2
Enter patient name : green
Enter age : 10
1.Create record lst
2.Insert new record
3.Display record list
4.Exit
Enter choice : 3
Name : red      Age : 13
Name : blue     Age : 11
Name : green    Age : 10

1.Create record lst
2.Insert new record
3.Display record list
4.Exit
Enter choice :
```

```c
/*Description: Implement a linked list to track medical tests for
patients. Operations:
Create a list of medical tests.
Insert a new test result.
Display all test results.*/
#include<stdio.h>
#include<stdlib.h>
#include<string.h>
struct test
{
    char name[10];
    struct test *next;
};
struct test *head=NULL,*ptr;
void create()
{
    int n;
    printf("Enter list size : ");
    scanf("%d",&n);
    ptr=head;
    for(int i=0;i<n;i++)
    {
        struct test * node=(struct test *)malloc(sizeof(struct test));
        strcpy(node->name,"\0");
        node->next=NULL;
        if(head==NULL)
        {
            head=node;
            ptr=node;
        }
        else
        {
            ptr->next=node;
            ptr=node;
        }
    }
}
void insert()
{
    char name[10];
```

```
    printf("Enter test name : ");
    scanf("%s",name);
    ptr=head;
    while(ptr!=0)
    {
        if(strcmp(ptr->name,"\0")==0)
        {
            strcpy(ptr->name,name);
            return;
        }
        ptr=ptr->next;
    }
}
void display()
{
    ptr=head;
    while(ptr!=NULL)
    {
        if(strcmp(ptr->name,"\0")!=0)
        printf("%s -> ",ptr->name);
        ptr=ptr->next;
    }
    printf("\n");
}
void main()
{
    int choice;
    do
    {
        printf("1.Create test lst\n");
        printf("2.Insert new test\n");
        printf("3.Display test list\n");
        printf("4.Exit\n");
        printf("Enter choice : ");
        scanf("%d",&choice);
        switch(choice)
        {
            case 1:
            create();
            break;
```

```
            case 2:
            insert();
            break;
            case 3:
            display();
            break;
            case 4:printf("Exiing..\n");
            break;
        }
    } while (choice!=4);

}
```

```
PS D:\projects\quest\C> cd "d:\projects\qu
1.Create test lst
2.Insert new test
3.Display test list
4.Exit
Enter choice : 1
Enter list size : 5
1.Create test lst
2.Insert new test
3.Display test list
4.Exit
Enter choice : 2
Enter test name : MRI
1.Create test lst
2.Insert new test
3.Display test list
4.Exit
Enter choice : 3
MRI ->
1.Create test lst
2.Insert new test
3.Display test list
4.Exit
Enter choice : 2
Enter test name : CTscan
1.Create test lst
2.Insert new test
3.Display test list
4.Exit
Enter choice : 3
MRI -> CTscan ->
1.Create test lst
2.Insert new test
3.Display test list
4.Exit
Enter choice :
```

```c
/*Description: Use a linked list to manage patient prescriptions.
Operations:
Create a prescription list.
Insert a new prescription.
Display all prescriptions.*/
#include<stdio.h>
#include<stdlib.h>
#include<string.h>
struct prescription
{
    char name[10];
    struct prescription *next;
};
struct prescription *head=NULL,*ptr;
void create()
{
    int n;
    printf("Enter list size : ");
    scanf("%d",&n);
    ptr=head;
    for(int i=0;i<n;i++)
    {
        struct prescription * node=(struct prescription
*)malloc(sizeof(struct prescription));
        strcpy(node->name,"\0");
        node->next=NULL;
        if(head==NULL)
        {
            head=node;
            ptr=node;
        }
        else
        {
            ptr->next=node;
            ptr=node;
        }
    }
}
void insert()
{
```

```
    char name[10];
    printf("Enter medicine name : ");
    scanf("%s",name);
    ptr=head;
    while(ptr!=0)
    {
        if(strcmp(ptr->name,"\0")==0)
        {
            strcpy(ptr->name,name);
            return;
        }
        ptr=ptr->next;
    }
}
void display()
{
    ptr=head;
    while(ptr!=NULL)
    {
        if(strcmp(ptr->name,"\0")!=0)
        printf("%s -> ",ptr->name);
        ptr=ptr->next;
    }
    printf("\n");
}
void main()
{
    int choice;
    do
    {
        printf("1.Create prescription lst\n");
        printf("2.Insert new prescription\n");
        printf("3.Display prescription list\n");
        printf("4.Exit\n");
        printf("Enter choice : ");
        scanf("%d",&choice);
        switch(choice)
        {
            case 1:
            create();
```

```
            break;
            case 2:
            insert();
            break;
            case 3:
            display();
            break;
            case 4:printf("Exiing..\n");
            break;
        }
    } while (choice!=4);

}
```

```
PS D:\projects\quest\C> cd "d:\projects\quest\
1.Create prescription lst
2.Insert new prescription
3.Display prescription list
4.Exit
Enter choice : 1
Enter list size : 5
1.Create prescription lst
2.Insert new prescription
3.Display prescription list
4.Exit
Enter choice : 2
Enter medicine name : med1
1.Create prescription lst
2.Insert new prescription
3.Display prescription list
4.Exit
Enter choice : 2
Enter medicine name : med2
1.Create prescription lst
2.Insert new prescription
3.Display prescription list
4.Exit
Enter choice : 2
Enter medicine name : med3
1.Create prescription lst
2.Insert new prescription
3.Display prescription list
4.Exit
Enter choice : 3
med1 -> med2 -> med3 ->
1.Create prescription lst
2.Insert new prescription
3.Display prescription list
4.Exit
Enter choice :
```

```c
/*Description: Develop a linked list to manage the hospital staff roster.
Operations:
Create a staff roster.
Insert a new staff member into the roster.
Display the current staff roster.*/
#include<stdio.h>
#include<stdlib.h>
#include<string.h>
struct staff
{
    char name[10];
    int id;
    struct staff *next;
};
struct staff *head=NULL,*ptr;
void create()
{
    int n;
    printf("Enter roaster size : ");
    scanf("%d",&n);
    ptr=head;
    for(int i=0;i<n;i++)
    {
        struct staff * node=(struct staff *)malloc(sizeof(struct staff));
        strcpy(node->name,"\0");
        node->id=0;
        node->next=NULL;
        if(head==NULL)
        {
            head=node;
            ptr=node;
        }
        else
        {
            ptr->next=node;
            ptr=node;
        }
    }
}
void insert()
```

```
{
    char name[10];
    int n;
    printf("Enter staff name : ");
    scanf("%s",name);
    printf("Enter id : ");
    scanf("%d",&n);
    ptr=head;
    while(ptr!=0)
    {
        if(strcmp(ptr->name,"\0")==0)
        {
            strcpy(ptr->name,name);
            ptr->id=n;
            return;
        }
        ptr=ptr->next;
    }
}
void display()
{
    ptr=head;
    while(ptr!=NULL)
    {
        if(strcmp(ptr->name,"\0")!=0)
       {
        printf("Staff name : %s\n",ptr->name);
        printf("Staff id : %d\n",ptr->id);
        printf("\n");
       }
        ptr=ptr->next;
    }
    printf("\n");
}
void main()
{
    int choice;
    do
    {
        printf("1.Create staff roater \n");
```

```
        printf("2.Insert new staff\n");
        printf("3.Display staff roaster\n");
        printf("4.Exit\n");
        printf("Enter choice : ");
        scanf("%d",&choice);
        switch(choice)
        {
            case 1:
            create();
            break;
            case 2:
            insert();
            break;
            case 3:
            display();
            break;
            case 4:printf("Exiing..\n");
            break;
        }
    } while (choice!=4);

}
```

```
1.Create staff roater
2.Insert new staff
3.Display staff roaster
4.Exit
Enter choice : 2
Enter staff name : Sarah
Enter id : 1001
1.Create staff roater
2.Insert new staff
3.Display staff roaster
4.Exit
Enter choice : 2
Enter staff name : James
Enter id : 1002
1.Create staff roater
2.Insert new staff
3.Display staff roaster
4.Exit
Enter choice : 2
Enter staff name : Jessie
Enter id : 1003
1.Create staff roater
2.Insert new staff
3.Display staff roaster
4.Exit
Enter choice : 3
Staff name : Sarah
Staff id : 1001

Staff name : James
Staff id : 1002

Staff name : Jessie
Staff id : 1003


1.Create staff roater
2.Insert new staff
3.Display staff roaster
4.Exit
Enter choice :
```