```c
#include<stdio.h>
#include<malloc.h>
#include<string.h>

struct node
{
    int pid,quantity;
    float price;
    char name[50];
    int out;
    int count;
    struct node *next;
};

struct pop
{
    int pid,quantity;
    float price;
    char name[50];
    int out;
    int count;
};

struct pop obj;
struct node *head=NULL,*p,*last=NULL,*q=NULL;

int n,m,i;
int pid,quantity;
float price;
char name[50];
void create();
void display();
void sort();
void search();
struct node* insert();
void reverse();
void write();
void read();
void delete1();
void modify();
//void enter();
void graph();
int search_pid(int);
int count();
int search_name();
int nameser();


int main()
{
    system("color 9");
    int ch,ch1,ch2,m=0,m1=0,ph_no,emp,pid1,loc,x,choice;
    char str1[20],str2[20],str3[]="abcd",ch3,ch5,ch4,c_name[50],address[50];
    char password[100],c=' ',str[20];
    int i=0;
    printf("\n\n-----------------------------------------\n");
    printf("     WELCOME TO My Restaurant                  ");
    printf("\n-----------------------------------------\n");
    printf("\n1.Admin \n2.User \nEnter Your Choice:");
    scanf("%d",&ch);
    switch(ch)
    {
    case 1:
        do
        {
            //printf("\nNote:Use only lower case letters & digits\n");
        printf("\n Enter Name:");
        scanf("%s",str1);
        printf("\n Enter Password:");
        while(i<9)
        {
            str[i]=getch();
            c=str[i];
            if(c==13)
            break;
            else printf("*");
            i++;
        }

        str[i]='\0';
        i=0;
        strlwr(str);
        if((strcmp(str,str3))==0)
        {
```

```
85
86      do
87      {
88      printf("\n\n-----------------------------------------------\n");
89      printf("                    MENU                       ");
90      printf("\n-----------------------------------------------\n");
91      printf("\n1 Create\n2.Display\n3.Insert
    \n4.Sort\n5.Reverse\n6.Search\n7.Delete\n8.Graph\n9.Modify\n.10 Exit");
92      printf("\nEnter Your Choice:");
93      scanf("%d",&ch);
94      switch(ch)
95      {
96      case 1:system("CLS");
97      create();
98      break;
99      case 2:system("CLS");
100     display();
101     break;
102     case 3:system("CLS");
103     insert();
104     break;
105     //n=n+1;*/
106     case 4:system("CLS");
107     sort(head);
108     break;
109     case 5:system("CLS");
110     reverse();
111     break;
112     case 7:system("CLS");
113     delete1();
114     break;
115     case 6:system("CLS");
116     search();
117     break;
118     case 9:system("CLS");
119     modify();
120     break;
121
122     case 8:system("CLS");
123     graph();
124     break;
125     }
126     }while(ch!=10);
127 return 0;
128 }
129 else
130         {
131             printf("\nPASSWORD IS WRONG !!!");
132         }
133
134         printf("\nDo You Want To Continue Y OR N:");
135         scanf("%s",&ch3);
136         strupr(ch3);
137
138
139     }while(ch3=='Y');
140     return 0;
141     break;
142     case 2:
143           //display();
144
145
146           do
147     {
148     printf("\n\n-----------------------------------------------\n");
149     printf("                  Restaurant MENU                  ");
150     printf("\n-----------------------------------------------\n");
151     printf("\n\n1.Display\n2.place your order\n3.Cancel order\n4.Exit");
152     printf("\nEnter Your Choice:");
153     scanf("%d",&choice);
154     switch(choice)
155     {
156
157     case 1:system("CLS");
158     display();
159     break;
160     case 2://system("CLS");
161     order();
162     break;
163     case 3:system("CLS");
164     cancelorder();
165     break;
166
167     }
```

```
168         }while(choice!=4);
169         return 0;
170         break;
171       }
172   }
173
174
175
176
177   void create()
178   {
179       int f=0,f2=0,x=0,y=0,cnt,not_entered=1;
180       read();
181       printf("\n\n-----------------------------------------\n");
182       printf("                 Creating Data                 ");
183       printf("\n-----------------------------------------\n");
184       printf("\nEnter Number of Items:");
185       scanf("%d",&n);
186      // enter();
187       struct node *p;
188       int i=0;
189       p=last;
190       if(last==NULL)
191       {
192       i=1;
193       last=(struct node*)malloc(sizeof(struct node));
194       last->next=NULL;
195       //enter();
196       printf("Enter The DishID:");
197       while(1)
198       {
199       scanf("\t%d",&(x));
200       if(x<1)
201       {
202       printf("\nThe DishId Should be a positive non Zero Integer");
203       break;
204       }
205
206
207       f=search_pid(x);
208       if(f==0)
209       {
210           last->pid=x;
211           break;
212       }
213       else
214       {
215           printf("DishID Already Exist...Enter The DishID Again:");
216           f=0;
217       }
218       }
219         printf("Enter The Name:");
220       //while(not_entered)
221       {
222
223       scanf("\t%s",&(obj.name));
224       strupr(obj.name);
225
226       /*f=nameser();
227       if(f==0)
228       {
229           strcpy(last->name,obj.name);
230           not_entered=1;
231           break;
232           //printf("DishName Already Exist...Enter The DishName Again:");
233           //f=0;
234       }
235       else
236       {
237           printf("DishName Already Exist...Enter The DishName Again:");
238           f=0;
239           //strcpy(last->name,obj.name);
240           //not_entered=1;
241           //break;
242       }*/
243       }
244
245
246       //printf("\nEnter The DishID:");
247       //scanf("\t%d",&(last->pid));
248       //printf("Enter The Name:");
249       //scanf("\t%s",last->name);
250       printf("Enter The Quantity:");
251       scanf("\t%d",&(last->quantity));
```

```
252         printf("Enter The Price:");
253         scanf("\t%f",&(last->price));
254         last->out = 0;              //patient is in queue
255         cnt = count();
256         last->count  = cnt+1;
257
258         p=last;
259         head=last;
260         }//if
261         p=last;
262         for(;i<n;i++)
263         {
264
265         p->next=(struct node*)malloc(sizeof(struct node));
266         p=p->next;
267         last=p;
268        printf("\nEnter The DishID:");
269         while(1)
270         {
271         scanf("\t%d",&x);
272         if(x<1)
273         {
274             printf("\nThe DishId Should be a positive non Zero Integer");
275
276         }
277         f2 = search_pid(x);
278           if(f2 == 0)
279             {
280            p->pid = x;
281            break;
282           }
283           else{
284             printf("\nDishID Already Exists..Enter a New DishID:");
285             f2 = 0;
286
287            }
288
289         }
290
291          printf("\nEnter The Name:");
292         while(not_entered)
293         {
294
295         scanf("\t%s",&(obj.name));
296         strupr(obj.name);//Automatically stores data in Uppercase format
297         f2=nameser();
298         if(f2==0)
299         {
300             strcpy(p->name,obj.name);
301             not_entered=1;
302             break;
303             //printf("DishName Already Exist...Enter The DishName Again:");
304             //f2=0;
305         }
306        else
307         {
308             printf("DishName Already Exist...Enter The DishName Again:");
309             f2=0;
310             //strcpy(p->name,obj.name);
311             //not_entered=1;
312             //break;
313         }
314         }
315
316
317         printf("Enter The Quantity:");
318         scanf("%d",&(p->quantity));
319         printf("Enter The Price:");
320         scanf("%f",&(p->price));
321         p->out = 0;
322         cnt = count();
323         last->count  = cnt+1;
324
325         p->next=NULL;
326         }
327         write();
328
329         }
330
331         void order()
332         {
333             char ch;
334             int qut;
335              int tbill;
```

```
            int tqut;

           FILE *ne=fopen("newfile.txt","r");
      int pid,found=0;
      printf("\n\n-----------------------------------------------\n");
      printf("                  Place Your Order                  ");
      printf("\n-----------------------------------------------\n");
      p=head;


      printf("\nEnter The Dish ID To Order:");
      scanf("%d",&pid);
      if(pid<1)
      {
          printf("\nThe DishId Should be a positive non Zero Integer");

      }
      else{

      for(i=1;p!=NULL;i++)
      {
      if(p->pid==pid)
      {
      //printf("\nData Found At %dth Location",i);
      printf("\nDishID:%d \tName:%s  \tPrice:%f\n",p->pid,p->name,p->price);
      found=1;
      printf("\nPlease Enter Quantity of Item");
      scanf("%d",&qut);
      tbill=p->price*qut;
      printf("\nYour Total Bill  is----> %d",tbill);
      tqut=p->quantity-qut;
      p->quantity=tqut;
      write();
      printf("\n The remaning items are  %d",tqut);


      }


      p=p->next;

      }


      if(found==1);
      printf("\nYour Order Has been Placed Sucessfully!!!!!!");
      if(!found)
      {
      printf("\nNo Entry Found Corresponding To Your Data\n");
      }
      fclose(ne);
      }



      }

void display()

{
      read();
      int i;
      printf("\n\n-----------------------------------------------\n");
      printf("                  Displaying Data                    ");
      printf("\n-----------------------------------------------\n");
      p=head;
      if(p==NULL)
      {
      printf("\nList Is Empty!!!");
      }
      else
      {
      printf("\n List Is:");
      printf("\n\tDishID\tDishName\tQuantity\tRate\n");
         for(i=0;p!=NULL;i++)
         {
             printf("\n");
             printf("\t%d\t",p->pid);
             printf("\t%s\t",p->name);
            printf("\t%d\t",p->quantity);
             printf("\t%.2f\t",p->price);
             p=p->next;
         }

```

```
    }

}

struct node* insert()
{
    int f=0,x=0,not_entered=1,cnt;
    read();
    printf("\n\n-----------------------------------------------\n");
    printf("                Inserting Data                  ");
    printf("\n-----------------------------------------------\n");
    p=head;
    q=head;

    int i,loc,pid,quantity;
    float price;
    char name[50];
    printf("\nEnter The New Position:");
    scanf("%d",&loc);

    printf("Enter New DishID:");
    while(1){
      scanf("\t%d",&x);
      if(x<1)
        printf("\nThe DishId Should be a positive non Zero Integer");

      f = search_pid(x);
      if(f == 0){
       pid = x;
       break;
      }
      else{
        printf("\nDish ID already exists..Enter a new one ");
        f=0;
       }
       fflush(stdin);
    }

    printf("Enter The Name:");
    while(not_entered)
    {

    scanf("\t%s",(obj.name));
    strupr(obj.name);

    f=nameser();
    if(f==0)
    {
        strcmp(last->name,obj.name)==0;
        not_entered=1;
        break;
    }
    else
    {
        printf("DishName Already Exist...Enter The DishName Again:");
        f=0;
    }
    }


    printf("Enter New Quantity:");
    scanf("%d",&quantity);
    printf("Enter The New Price:");
    scanf("%f",&price);


    p=(struct node*)malloc(sizeof(struct node));
    p->pid=pid;
    strcpy(p->name,obj.name);
    p->quantity=quantity;
    p->price=price;
    p->out = 0;
    cnt = count();
    p->count  = cnt+1;
    p->next=NULL;
    if(loc==1)
    {
        p->next=head;
    head=p;
        write();
        return(p);
    }
    q=head;
    for(i=1;i<loc-1;i++)
```

```
    {

        if(q!=NULL)
        {
            q=q->next;
        }
    }
    p->next=q->next;
    q->next=p;
    n=n+1;
    write();
    return(head);
}

void modify()
{
    int f=0,x=0,not_entered=1,cnt;
    int i=1,ch,choice=1,loc=0;
    printf("\n\n-----------------------------------------------\n");
    printf("                 Modifying Data                  ");
    printf("\n-----------------------------------------------\n");
    read();
    printf("\nEnter The Location You Want To Modify:");
    scanf("%d",&loc);
    p=head;
    for(i=1;i<=loc;i++)
    {
    if(loc==i)
    {
    do{
    printf("\n0.Exit");
    printf("\t1.DishID:%d",p->pid);
    printf("\t2.Name:%s",p->name);
    printf("\t3.Quantity:%d",p->quantity);
    printf("\t4.Price:%.2f\n",p->price);
    printf("\nEnter Your Choice To Edit:");
    scanf("%d",&ch);
    switch(ch)
    {
    case 1:

    printf("\nEnter The New DishID:");
    while(1){
      scanf("%d",&x);
      if(x<1)
        printf("\nThe DishId Should be a positive non Zero Integer");
      f = search_pid(x);
      if(f == 0){
       p->pid = x;
       break;
      }
      else{
        printf("Dish ID already exists..Enter a new one ");
        f=0;
       }
    }
    break;
    case 2:
        printf("Enter The Name:");
    while(not_entered)
    {

    scanf("\t%s",(obj.name));
    strupr(obj.name);

    f=nameser();
    if(f==0)
    {
        strcpy(last->name,obj.name);
        not_entered=1;
        break;
    }
    else
    {
        printf("DishName Already Exist...Enter The DishName Again:");
        f=0;

    }
    }


    break;
    case 3:

```

```
588         printf("Enter The New Quantity:");
589         scanf("%d",&p->quantity);
590         break;
591         case 4:
592
593         printf("Enter New Price:");
594         scanf("%f",&p->price);
595         break;
596         }//switch
597         }while(ch!=0);
598
599         }//if
600         if(p->next==NULL)
601         {
602         break;
603         }//if
604         p=p->next;
605
606         }//for
607
608         write();
609     }
610
611     void sort()
612         {
613         int k;
614         printf("\n\n-----------------------------------------------\n");
615         printf("                  Sorting Data                   ");
616         printf("\n-----------------------------------------------\n");
617         read();
618         int temp;
619         struct node *i,*j;
620         for(i=head;i!=NULL;i=i->next)
621         {
622         for(j=i->next;j!=NULL;j=j->next)
623         {
624         if((i->pid)>(j->pid))              //write integer variable only in place of data
625         {
626         obj.pid=i->pid;
627         obj.quantity=i->quantity;
628         obj.price=i->price;
629
630         i->pid=j->pid;
631         i->quantity=j->quantity;
632         i->price=j->price;
633
634         j->pid=obj.pid;
635         j->quantity=obj.quantity;
636         j->price=obj.price;
637         for(k=0;k<sizeof(obj.name);k++)
638         {
639         obj.name[k]=i->name[k];
640         i->name[k]=j->name[k];
641         j->name[k]=obj.name[k];
642         }
643         }//if
644         }//j for
645         }//i for
646
647         write();
648         }//void sort
649
650
651     void reverse()
652         {
653         read();
654         printf("\n\n-----------------------------------------------\n");
655         printf("                 Reversing Data                  ");
656         printf("\n-----------------------------------------------\n");
657
658         struct node *prenode,*currnode;
659         if(head!=NULL)
660         {
661         prenode=head;
662         currnode=head->next;
663         prenode->next=NULL;
664         }
665         while(head!=NULL)
666         {
667         head=currnode->next;
668         currnode->next=prenode;
669         prenode=currnode;
670         currnode=head;
671         }
```

```
672       head=prenode;
673
674       write();
675
676       }//void reverse
677
678   void delete1()
679   {
680       read();
681       printf("\n\n-----------------------------------------------\n");
682       printf("                  Deleting Data                    ");
683       printf("\n-----------------------------------------------\n");
684       int loc,i;
685       printf("\nEnter The Location To Delete:");
686       scanf("%d",&loc);
687       printf("\nNumber %d is Being Deleted..Please Wait",loc);
688
689       if(p!=NULL)
690       {
691       q=head;
692       p=q->next;
693
694       for(i=1;i<=loc;)
695       {
696       if(i==2)
697       {
698       q=head;
699       p=q->next;
700       }
701       if(i==loc&&i==1)
702       {
703       head=head->next;
704       printf("\nDishID:%d",q->pid);
705       free(q);
706       write();
707       printf("\nfree q");
708       break;
709       }
710       else
711       {
712       if(i==loc)
713       {
714       q->next=p->next;
715       printf("\nq->pid:%d",p->pid);
716       write();
717       free(p);
718       printf("\nfree p");
719       }//if
720       }//else
721       printf("\nIncremented");
722       p=p->next;
723       q=q->next;
724       i++;
725       }//for
726       p=head;
727       q=head;
728       }
729       else
730       {
731       printf("\nSorry,The List Is Empty!!!");
732       }
733       }//void delete
734
735   void search()
736       {
737           FILE *ne=fopen("newfile.txt","r");
738       int pid,found=0;
739       printf("\n\n-----------------------------------------------\n");
740       printf("                   Searching Data                    ");
741       printf("\n-----------------------------------------------\n");
742       p=head;
743
744       printf("\nEnter The Dish ID To Search:");
745       scanf("%d",&pid);
746       if(pid<1)
747       printf("\nThe DishId Should be a positive non Zero Integer");
748
749       for(i=1;p!=NULL;i++)
750       {
751       if(p->pid==pid)
752       {
753       printf("\nData Found At %dth Location",i);
754       printf("\nDishID:%d \tName:%s \tQuantity:%d
      \tPrice:%f\n",p->pid,p->name,p->quantity,p->price);
```

```
755         found=1;
756         }
757         p=p->next;
758         }
759         if(!found)
760         {
761         printf("\nNo Entry Found Corresponding To Your Data\n");
762         }
763         fclose(ne);
764
765
766         }
767
768     void write()
769         {
770
771         FILE *ne=fopen("newfile.txt","w");
772         int i;
773         struct node *temp;
774         temp=head;
775         if(temp==NULL)
776         {
777             printf("\nList Is Empty!!!");
778         }
779         else
780         {
781         printf("\nList Is Being Saved!!!");
782         for(i=0;temp!=NULL;i++)
783         {
784         fprintf(ne,"%d %s %d %f \n",temp->pid,temp->name,temp->quantity,temp->price);
785
786         temp=temp->next;
787         }//for
788         printf("\nDone\n");
789
790         }//else
791
792         fclose(ne);
793         }
794
795     void read()
796         {
797         int i,filempty=0;
798         FILE *infile=fopen("newfile.txt","r");
799         p=head;
800         fseek(infile,0,SEEK_END);
801
802         if(p==NULL)
803         {
804         p=(struct node*)malloc(sizeof(struct node));
805         head=p;
806         }
807
808         int len=(int)ftell(infile);
809         if(len<=0)
810         {
811         filempty=1;
812         printf("\nFile Empty!!!");
813         p=NULL;
814         head=p;
815         }
816
817         if(filempty==0)
818         {
819         rewind(infile);
820         while(fscanf(infile,"%d %s %d %f \n",&p->pid,p->name,&p->quantity,&p->price))
821         {
822         if(feof(infile))
823         {
824         break;
825         }
826         p->next=(struct node*)malloc(sizeof(struct node));
827         p=p->next;
828         last=p;
829         p->next=NULL;
830         }
831         }
832         }
833
834     void graph()
835     {
836         int j;
837
838         int value;
```

```
839         float height=0;
840         int length;
841         read();
842         printf("\n\n-----------------------------------------------\n");
843         printf("                   Graph                       ");
844         printf("\n-----------------------------------------------\n");
845         read();
846         p=head;
847         for(i=1;p!=NULL;i++)
848         {
849         if(height<p->quantity)
850         {
851         height=p->quantity;
852         }
853         p=p->next;
854         }
855         length=40/i;
856         p=head;
857
858         for(i=(int)height+10;i>=0;i--)
859         {
860         printf("\n*");
861         p=head;
862         for(;p!=NULL;)
863         {
864         value=(int)p->quantity;
865         if(i<=value)
866         {
867         printf("\t*");
868         }
869         else{printf("\t");}
870         p=p->next;
871         }
872         }
873         p=head;
874         printf("\n\n");
875         printf("0");
876         for(j=1;p!=NULL;j++)
877         {
878         printf("\t%d",j);
879         p=p->next;
880         }
881         printf("\n\n");
882         p=head;
883         for(;p!=NULL;)
884         {
885         printf("%s(%.2d)\t",p->name,p->quantity);
886         p=p->next;
887         }
888
889     }
890
891
892
893     int search_pid(int pid){
894         struct node* x = head;
895         int i=0;
896         if(x==NULL)
897         {
898           return 0;
899         }else{
900             for(i=0;x!=last;i++)
901             {
902               if(x->pid == pid)
903                 return 1;
904               x=x->next;
905
906
907             }
908
909         }
910         return 0;
911     }
912
913     int nameser()
914     {
915
916     struct node*y=head;
917     //y=head;
918         for(i=0;y!=last;i++)
919         {
920             if(!strcmp(y->name,obj.name))
921             {
922             return 1;
```

```
923
924             }
925             y=y->next;
926         }
927
928     return 0;
929     }
930
931
932     int count(){
933         int i=0,max;
934         struct node* x =head;
935
936         if(x==NULL)
937         {
938           return 0;
939         }else{
940             max = x->count;
941             for(i=0;x!=last;i++)
942             {
943               if(x->count > max)
944                max = x->count;
945               x=x->next;
946
947             }
948
949         }
950         return max;
951     }
952
953         void cancelorder()
954         {
955             char ch;
956             FILE *ne=fopen("newfile.txt","r");
957         int pid,found=0;
958         printf("\n\n-----------------------------------------------\n");
959         printf("                  Canceling  Your Order                    ");
960         printf("\n-----------------------------------------------\n");
961         p=head;
962
963
964         printf("\nEnter The Dish ID To Cancel Order:");
965         scanf("%d",&pid);
966
967         for(i=1;p!=NULL;i++)
968         {
969         if(p->pid==pid)
970         {
971         printf("\nDishID:%d \tName:%s  \tPrice:%f\n",p->pid,p->name,p->price);
972         found=1;
973         }
974         p=p->next;
975         }
976         if(found==1);
977         printf("\nYour Order Has been   Canceled Sucessfully!!!!!!");
978         if(!found)
979         {
980         printf("\nNo Entry Found Corresponding To Your Data\n");
981         }
982         fclose(ne);
983
984
985
986         }
987
988
989
```