

```

1  #include<stdio.h>
2  #include<malloc.h>
3  #include<string.h>
4
5  struct node
6  {
7      int pid,quantity;
8      float price;
9      char name[50];
10     int out;
11     int count;
12     struct node *next;
13 };
14
15 struct pop
16 {
17     int pid,quantity;
18     float price;
19     char name[50];
20     int out;
21     int count;
22 };
23
24 struct pop obj;
25 struct node *head=NULL,*p,*last=NULL,*q=NULL;
26
27 int n,m,i;
28 int pid,quantity;
29 float price;
30 char name[50];
31 void create();
32 void display();
33 void sort();
34 void search();
35 struct node* insert();
36 void reverse();
37 void write();
38 void read();
39 void delete1();
40 void modify();
41 //void enter();
42 void graph();
43 int search_pid(int);
44 void queue();
45 int count();
46 int search_name();
47 int nameser();
48
49
50 int main()
51 {
52     system("color 9");
53     int ch,ch1,ch2,m=0,m1=0,ph_no,emp,pid1,loc,x,choice;
54     char str1[20],str2[20],str3[]="abcd",ch3,ch5,ch4,c_name[50],address[50];
55     char password[100],c=' ',str[20];
56     int i=0;
57     printf("\n\n-----\n");
58     printf("        WELCOME TO My Restaurant        ");
59     printf("\n-----\n");
60     printf("\n1.Admin \n2.User \nEnter Your Choice:");
61     scanf("%d",&ch);
62     switch(ch)
63     {
64     case 1:
65         do
66         {
67             //printf("\nNote:Use only lower case letters & digits\n");
68             printf("\n Enter Name:");
69             scanf("%s",str1);
70             printf("\n Enter Password:");
71             while(i<9)
72             {
73                 str[i]=getch();
74                 c=str[i];
75                 if(c==13)
76                     break;
77                 else printf("*");
78                 i++;
79             }
80
81             str[i]='\0';
82             i=0;
83             strlwr(str);
84             if((strcmp(str,str3))==0)

```

```

85         {
86
87         do
88         {
89             printf("\n\n-----\n");
90             printf("                MENU                ");
91             printf("\n\n-----\n");
92             printf("\n1 Create\n2.Display\n3.Insert
\n4.Sort\n5.Reverse\n6.Search\n7.Delete\n8.Graph\n9.Modify\n10 Exit");
93             printf("\nEnter Your Choice:");
94             scanf("%d",&ch);
95             switch(ch)
96             {
97                 case 1:system("CLS");
98                 create();
99                 break;
100                case 2:system("CLS");
101                display();
102                break;
103                case 3:system("CLS");
104                insert();
105                break;
106                //n=n+1;*/
107                case 4:system("CLS");
108                sort(head);
109                break;
110                case 5:system("CLS");
111                reverse();
112                break;
113                /*case 6:system("CLS");
114                write();
115                break;
116                case 7:system("CLS");
117                read();
118                break;*/
119                case 7:system("CLS");
120                deletel();
121                break;
122                case 6:system("CLS");
123                search();
124                break;
125                case 9:system("CLS");
126                modify();
127                break;
128
129                case 8:system("CLS");
130                graph();
131                break;
132            }
133        }while(ch!=10);
134        return 0;
135    }
136    else
137    {
138        printf("\nPASSWORD IS WRONG !!!");
139    }
140    printf("\nDo You Want To Continue Y OR N:");
141    scanf("%s",&ch3);
142    strlwr(ch3);
143    }while(ch3=='y');
144    return 0;
145    break;
146    case 2:
147        //display();
148
149
150        do
151        {
152            printf("\n\n-----\n");
153            printf("                Restaurant MENU                ");
154            printf("\n\n-----\n");
155            printf("\n\n1.Display\n2.place your order\n3.Cancel order\n4.Exit");
156            printf("\nEnter Your Choice:");
157            scanf("%d",&choice);
158            switch(choice)
159            {
160
161                case 1:system("CLS");
162                display();
163                break;
164                case 2://
165                system("CLS");
166                order();
167                break;

```

```

168     case 3:system("CLS");
169     cancelorder();
170     break;
171 }
172 //printf("\nDo You Want To See Again Data Y OR N:");
173 //scanf("%s",&ch4);
174 }while(choice!=4);
175 return 0;
176 break;
177 }
178 }
179 }
180
181
182 /* void enter()
183 {
184     printf("\nEnter The DishID:");
185     scanf("%t%d",&(pid));
186     printf("Enter The Name:");
187     scanf("%t%s",name);
188     printf("Enter The Quantity:");
189     scanf("%t%d",&(quantity));
190     printf("Enter The Price:");
191     scanf("%t%f",&(price));
192
193     q=head;
194     for(;q!=NULL;)
195     {
196         if(pid==q->pid&&strcmp(name,q->name))
197         {
198             printf("Record with same name or pid already exist.Please enter again");
199             enter();
200         }
201
202         q=q->next;
203     }
204 }*/
205
206 void create()
207 {
208     int f=0,f2=0,x=0,y=0,cnt,not_entered=1;
209     read();
210     printf("\n\n-----\n");
211     printf("                Creating Data                ");
212     printf("\n\n-----\n");
213     printf("\nEnter Number of Items:");
214     scanf("%d",&n);
215     // enter();
216     struct node *p;
217     int i=0;
218     p=last;
219     if(last==NULL)
220     {
221         i=1;
222         last=(struct node*)malloc(sizeof(struct node));
223         last->next=NULL;
224         //enter();
225         printf("Enter The DishID:");
226         while(1)
227         {
228             scanf("%t%d",&(x));
229             f=search_pid(x);
230             if(f==0)
231             {
232                 last->pid=x;
233                 break;
234             }
235             else
236             {
237                 printf("DishID Already Exist...Enter The DishID Again:");
238                 f=0;
239             }
240         }
241         printf("Enter The Name:");
242         //while(not_entered)
243         {
244             scanf("%t%s",&(obj.name));
245             /*f=nameser();
246             if(f==0)
247             {
248                 strcpy(last->name,obj.name);
249                 not_entered=1;
250                 break;

```

```

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

```

```

336     scanf("%d",&(p->quantity));
337     printf("Enter The Price:");
338     scanf("%f",&(p->price));
339     p->out = 0; // patient is in queue
340     cnt = count();
341     last->count = cnt+1;
342     //enter();
343     p->next=NULL;
344     } //i for
345     write();
346
347 }
348
349
350 void display()
351 {
352
353     read();
354     int i;
355     printf("\n\n-----\n");
356     printf("                Displaying Data                ");
357     printf("\n-----\n");
358     p=head;
359     if(p==NULL)
360     {
361         printf("\nList Is Empty!!!");
362     }
363     else
364     {
365         printf("\n List Is:");
366         printf("\n\tDishID\tDishName\tQuantity\tRate\n");
367         for(i=0;p!=NULL;i++)
368         {
369             printf("\n");
370             printf("\t%d\t",p->pid);
371             printf("\t%s\t",p->name);
372             printf("\t%d\t",p->quantity);
373             printf("\t%.2f\t",p->price);
374             p=p->next;
375         } //for
376
377     } //else
378
379 }
380
381 struct node* insert()
382 {
383     int f=0,x=0,not_entered=1,cnt;
384     read();
385     printf("\n\n-----\n");
386     printf("                Inserting Data                ");
387     printf("\n-----\n");
388     p=head;
389     q=head;
390
391     int i,loc,pid,quantity;
392     float price;
393     char name[50];
394     printf("\nEnter The New Position:");
395     scanf("%d",&loc);
396     /*if(strcmp(pid1,pid)
397     {
398         printf("These DishID Already Exists!!!");
399     }
400     else
401     {
402         */
403     printf("Enter New DishID:");
404     while(1){
405         scanf("\t%d",&x);
406         f = search_pid(x);
407         if(f == 0){
408             pid = x;
409             break;
410         }
411         else{
412             printf("\nDish ID already exists..Enter a new one ");
413             f=0;
414         }
415         fflush(stdin);
416     }
417
418     printf("Enter The Name:");
419     while(not_entered)

```

```

420     {
421
422         scanf("\t%s", (obj.name));
423         f=nameser();
424         if(f==0)
425         {
426             strcmp(last->name,obj.name)==0;
427             not_entered=1;
428             break;
429             //printf("DishName Already Exist...Enter The DishName Again:");
430             //f=0;
431         }
432         else
433         {
434             printf("DishName Already Exist...Enter The DishName Again:");
435             f=0;
436             //strcpy(last->name,obj.name);
437             //not_entered=1;
438             //break;
439         }
440     }
441
442     /*printf("Enter New DishID:");
443     scanf("%d",&pid);
444     printf("Enter New Name:");
445     scanf("%s",name);*/
446     printf("Enter New Quantity:");
447     scanf("%d",&quantity);
448     printf("Enter The New Price:");
449     scanf("%f",&price);
450     //enter();
451
452     p=(struct node*)malloc(sizeof(struct node));
453     p->pid=pid;
454     strcpy(p->name,obj.name);
455     p->quantity=quantity;
456     p->price=price;
457     p->out = 0;        //patient is in queue
458     cnt = count();
459     p->count = cnt+1;
460     p->next=NULL;
461     if(loc==1)
462     {
463         p->next=head;
464         head=p;
465         write();
466         return(p);
467     }
468     q=head;
469     for(i=1;i<loc-1;i++)
470     {
471
472         if(q!=NULL)
473         {
474             q=q->next;
475         }
476     }
477     p->next=q->next;
478     q->next=p;
479     n=n+1;
480     write();
481     return(head);
482 }
483
484 void modify()
485 {
486     int f=0,x=0,not_entered=1,cnt;
487     int i=1,ch,choice=1,loc=0;
488     printf("\n\n-----\n");
489     printf("                          Modifying Data                          ");
490     printf("\n-----\n");
491     read();
492     printf("\nEnter The Location You Want To Modify:");
493     scanf("%d",&loc);
494     p=head;
495     for(i=1;i<=loc;i++)
496     {
497         if(loc==i)
498         {
499             do{
500                 printf("\n0.Exit");
501                 printf("\t1.DishID:%d",p->pid);
502                 printf("\t2.Name:%s",p->name);
503                 printf("\t3.Quantity:%d",p->quantity);

```

```

504     printf("\t4.Price:%.2f\n",p->price);
505     printf("\nEnter Your Choice To Edit:");
506     scanf("%d",&ch);
507     switch(ch)
508     {
509     case 1:
510
511         printf("\nEnter The New DishID:");
512         while(1){
513             scanf("%d",&x);
514             f = search_pid(x);
515             if(f == 0){
516                 p->pid = x;
517                 break;
518             }
519             else{
520                 printf("Dish ID already exists..Enter a new one ");
521                 f=0;
522             }
523         }
524         //scanf("%d",&p->pid);
525         //enter();
526         break;
527     case 2:
528         printf("Enter The Name:");
529         while(not_entered)
530         {
531
532             scanf("\t%s", (obj.name));
533             f=nameser();
534             if(f==0)
535             {
536                 strcpy(last->name,obj.name);
537                 not_entered=1;
538                 break;
539                 //printf("DishName Already Exist...Enter The DishName Again:");
540                 //f=0;
541             }
542             else
543             {
544                 printf("DishName Already Exist...Enter The DishName Again:");
545                 f=0;
546                 //strcpy(last->name,obj.name);
547                 //not_entered=1;
548                 //break;
549             }
550         }
551
552         //enter();
553         break;
554     case 3:
555
556         printf("Enter The New Quantity:");
557         scanf("%d",&p->quantity);
558         //enter();
559         break;
560     case 4:
561
562         printf("Enter New Price:");
563         scanf("%f",&p->price);
564         //enter();
565         break;
566     } //switch
567     } while(ch!=0);
568
569     } //if
570     if(p->next==NULL)
571     {
572         break;
573     } //if
574     p=p->next;
575
576     } //for
577
578     write();
579 }
580
581 void sort()
582 {
583     int k;
584     printf("\n\n-----\n");
585     printf("                Sorting Data                ");
586     printf("\n\n-----\n");
587     read();

```

```

588     int temp;
589     struct node *i,*j;
590     for (i=head; i!=NULL; i=i->next)
591     {
592         for (j=i->next; j!=NULL; j=j->next)
593         {
594             if ((i->pid)>(j->pid)) //write integer variable only in place of data
595             {
596                 obj.pid=i->pid;
597                 obj.quantity=i->quantity;
598                 obj.price=i->price;
599
600                 i->pid=j->pid;
601                 i->quantity=j->quantity;
602                 i->price=j->price;
603
604                 j->pid=obj.pid;
605                 j->quantity=obj.quantity;
606                 j->price=obj.price;
607                 for (k=0; k<sizeof(obj.name); k++)
608                 {
609                     obj.name[k]=i->name[k];
610                     i->name[k]=j->name[k];
611                     j->name[k]=obj.name[k];
612                 }
613             } //if
614         } //j for
615     } //i for
616
617     write();
618 } //void sort
619
620
621 void reverse()
622 {
623     read();
624     printf("\n\n-----\n");
625     printf("                Reversing Data                ");
626     printf("\n-----\n");
627
628     struct node *prenode,*currnode;
629     if (head!=NULL)
630     {
631         prenode=head;
632         currnode=head->next;
633         prenode->next=NULL;
634     }
635     while (head!=NULL)
636     {
637         head=currnode->next;
638         currnode->next=prenode;
639         prenode=currnode;
640         currnode=head;
641     }
642     head=prenode;
643
644     write();
645
646 } //void reverse
647
648 void delete1()
649 {
650     read();
651     printf("\n\n-----\n");
652     printf("                Deleting Data                ");
653     printf("\n-----\n");
654     int loc,i;
655     printf("\nEnter The Location To Delete:");
656     scanf("%d",&loc);
657     printf("\nNumber %d is Being Deleted..Please Wait",loc);
658
659     if (p!=NULL)
660     {
661         q=head;
662         p=q->next;
663
664         for (i=1; i<=loc; i++)
665         {
666             if (i==2)
667             {
668                 q=head;
669                 p=q->next;
670             }
671             if (i==loc&& i==1)

```



```

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

```

```

755     } //for
756     printf("\nDone\n");
757
758     } //else
759
760     fclose(ne);
761 }
762
763 void read()
764 {
765     int i, fileempty=0;
766     FILE *infile=fopen("newfile.txt", "r");
767     p=head;
768     fseek(infile, 0, SEEK_END);
769
770     if (p==NULL)
771     {
772         p=(struct node*)malloc(sizeof(struct node));
773         head=p;
774     }
775
776     int len=(int)ftell(infile);
777     if (len<=0)
778     {
779         fileempty=1;
780         printf("\nFile Empty!!!");
781         p=NULL;
782         head=p;
783     }
784
785     if (fileempty==0)
786     {
787         rewind(infile);
788         while (fscanf(infile, "%d %s %d %f \n", &p->pid, p->name, &p->quantity, &p->price))
789         {
790             if (feof(infile))
791             {
792                 break;
793             }
794             p->next=(struct node*)malloc(sizeof(struct node));
795             p=p->next;
796             last=p;
797             p->next=NULL;
798         }
799     }
800 }
801
802 void graph()
803 {
804     int j;
805
806     int value;
807     float height=0;
808     int length;
809     read();
810     printf("\n\n-----\n");
811     printf("                                Graph                                ");
812     printf("\n-----\n");
813     read();
814     p=head;
815     for (i=1; p!=NULL; i++)
816     {
817         if (height<p->quantity)
818         {
819             height=p->quantity;
820         }
821         p=p->next;
822     }
823     length=40/i;
824     p=head;
825
826     for (i=(int)height+10; i>=0; i--)
827     {
828         printf("\n*");
829         p=head;
830         for (; p!=NULL;)
831         {
832             value=(int)p->quantity;
833             if (i<=value)
834             {
835                 printf("\t*");
836             }
837             else {printf("\t");}
838             p=p->next;

```

```

839     }
840     }
841     p=head;
842     printf("\n\n");
843     printf("0");
844     for(j=1;p!=NULL;j++)
845     {
846         printf("\t%d",j);
847         p=p->next;
848     }
849     printf("\n\n");
850     p=head;
851     for(;p!=NULL;)
852     {
853         printf("%s(%.2d)\t",p->name,p->quantity);
854         p=p->next;
855     }
856
857 }
858
859 void queue()
860 {
861     int i,max;
862     printf("\n\n-----\n");
863     printf("                                Queue                                ");
864     printf("\n\n-----\n");
865     read();
866     struct node* x = head;
867     max = count();
868     if(x==NULL){
869         printf("\n No Data!!!");
870         return;
871     }else
872     {
873         for(i=1;i<= max; i++)
874         {
875             x = head;
876             while( x!= NULL)
877             {
878                 if( (x->count == i) && (x->out ==0) )
879                 {
880                     printf("\n DishID : %d",x->pid);
881                     printf("\n DishName : %s",x->name);
882                     x->out = 1;
883                     write();
884                     return;
885                 }
886                 x= x->next;
887             }
888         }
889         printf("\n No Data!!!");
890         return ;
891     }
892
893 }
894
895 int search_pid(int pid){
896     struct node* x = head;
897     int i=0;
898     if(x==NULL)
899     {
900         return 0;
901     }else{
902         for(i=0;x!=last;i++)
903         {
904             if(x->pid == pid)
905                 return 1;
906             x=x->next;
907         }
908     }
909
910 }
911
912 return 0;
913 }
914
915 int nameser()
916 {
917     struct node*y=head;
918     //y=head;
919     for(i=0;y!=last;i++)
920     {
921         if(!strcmp(y->name,obj.name))
922

```

```

923     {
924         return 1;
925     }
926     }
927     y=y->next;
928 }
929
930 return 0;
931 }
932
933
934 int count(){
935     int i=0,max;
936     struct node* x =head;
937
938     if(x==NULL)
939     {
940         return 0;
941     }else{
942         max = x->count;
943         for(i=0;x!=last;i++)
944         {
945             if(x->count > max)
946                 max = x->count;
947             x=x->next;
948         }
949     }
950
951 }
952 return max;
953 }
954 void order()
955 {
956     char ch;
957     int qut;
958     int tbill;
959
960     FILE *ne=fopen("newfile.txt","r");
961     int pid,found=0;
962     printf("\n\n-----\n");
963     printf("                Place Your Order                ");
964     printf("\n-----\n");
965     p=head;
966
967
968     printf("\nEnter The Dish ID To Order:");
969     scanf("%d",&pid);
970
971     for(i=1;p!=NULL;i++)
972     {
973         if(p->pid==pid)
974         {
975             //printf("\nData Found At %dth Location",i);
976             printf("\nDishID:%d \tName:%s \tPrice:%f\n",p->pid,p->name,p->price);
977             found=1;
978             printf("\nPlease Enter Quantity of Item");
979             scanf("%d",&qut);
980             tbill=p->price*qut;
981             printf("\nYour Total Bill is----> %d",tbill);
982
983         }
984         p=p->next;
985     }
986
987
988     if(found==1);
989     printf("\nYour Order Has been Placed Sucessfully!!!!!!");
990     if(!found)
991     {
992         printf("\nNo Entry Found Corresponding To Your Data\n");
993     }
994     fclose(ne);
995
996
997
998 }
999 void cancelorder()
1000 {
1001     char ch;
1002     FILE *ne=fopen("newfile.txt","r");
1003     int pid,found=0;
1004     printf("\n\n-----\n");
1005     printf("                Canceling Your Order                ");
1006     printf("\n-----\n");

```

```

1007     p=head;
1008
1009
1010     printf("\nEnter The Dish ID To Cancel Order:");
1011     scanf("%d",&pid);
1012
1013     for (i=1;p!=NULL;i++)
1014     {
1015         if (p->pid==pid)
1016         {
1017             //printf("\nData Found At %dth Location",i);
1018             printf("\nDishID:%d \tName:%s \tPrice:%f\n",p->pid,p->name,p->price);
1019             found=1;
1020         }
1021         p=p->next;
1022     }
1023     if (found==1);
1024     printf("\nYour Order Has been Canceled Sucessfully!!!!");
1025     if (!found)
1026     {
1027         printf("\nNo Entry Found Corresponding To Your Data\n");
1028     }
1029     fclose(ne);
1030
1031
1032
1033     }
1034

```