

assignment-9

```
1.  input:
2.  #include<stdio.h>
3.  #include<stdlib.h>
4.  #include<string.h>
5.
6.  struct stuff
7.  {
8.      char eng_stuff[20];
9.      int price;
10.     int id;
11.     int quantity;
12.
13.     struct stuff *link;
14. };
15.
16. struct stuff *create(struct stuff *start);
17. void display (struct stuff *start);
18. struct stuff *insert(struct stuff *start);
19. struct stuff *del(struct stuff *start,char item[]);
20. struct stuff *sort (struct stuff *start);
21. void *search(struct stuff *start,char item[]);
22. struct stuff *modify(struct stuff *start,char item[]);
23. struct stuff *front=NULL;
24. struct stuff *rear=NULL;
25. void create_queue();
26. void del_queue();
27. void display_queue(struct stuff *front);
28. void graph(struct stuff *start,char [20]);
29. struct stuff *p,*q,*temp;
30.
31. int i,n;
32. void main()
33. {
34.     printf("\n\t\t\t\t*****WELCOME TO NOBEL_OF_NETWORK ENG_STUFF*****\n");
```

```

35.  char item[20];
36.  int ch;
37.  struct stuff *start=NULL;
38.  while(1)
39.  {
40.      printf("\nEnter choice-\n 1) to create\n 2) to create using queue\n 3) to display\n 4) to display
        using queue\n 5) to insert\n 6) to delete\n");
41.      printf(" 7) to delete using queue\n 8) to sort by price\n 9) to search\n 10) to modify:\n 11) To
        check in Profit or in Loss\t:");
42.      scanf("%d",&ch);
43.      switch (ch)
44.      {
45.          case 1:
46.              start=create(start);
47.              break;
48.          case 2:
49.              create_queue();
50.              break;
51.          case 3:
52.              display(start);
53.              break;
54.          case 4:
55.              display_queue(front);
56.              break;
57.          case 5:
58.              start=insert(start);
59.              break;
60.          case 6:
61.              printf("Enter stuff you want to delete:");
62.              scanf("%s",item);
63.              start=del(start,item);
64.              break;
65.          case 7:
66.              del_queue();
67.              break;
68.          case 8:

```

```

69.     sort(start);
70.     break;
71. case 9:
72.     printf("Enter stuff you want to search:");
73.     scanf("%s",item);
74.     search(start,item);
75.     break;
76. case 10:
77.     printf("Enter stuff you want to modify:");
78.     scanf("%s",item);
79.     start=modify(start,item);
80.     break;
81. case 11:
82.     printf("\n\t\t\t***|||*** represent PROFIT");
83.     printf("\n\n\t\t\t***|||*** represent LOSS");
84.     printf("\nEnter item for it's profit / loss graph:\t");
85.     scanf("%s",item);
86.     graph(start,item);
87.     break;
88. default:
89.     printf("error!");
90.
91. }
92. }
93. }
94. struct stuff *create(struct stuff* start)
95. {
96.     struct stuff *p,*temp;
97.     printf("Enter no of stuff:");
98.     scanf("%d",&n);
99.
100.     temp=start;
101.     for(i=0;i<n;i++)
102.     {
103.         temp=(struct stuff*)malloc(sizeof(struct stuff)*1);

```

```
104.         printf("\nEnter Name of eng_stuff %d:",i+1);
105.         scanf("%s",(temp->eng_stuff));
106.         printf("id of stuff:");
107.         scanf("%d",&(temp->id));
108.         printf("Quantity :");
109.         scanf("%d",&(temp->quantity));
110.         printf("Price of One eng_stuff:");
111.         scanf("%d",&(temp->price));
112.
113.         if(start==NULL)
114.         {
115.             start=temp;
116.             temp->link=NULL;
117.         }
118.     else
119.     {
120.         p=start;
121.         while(p->link!=NULL)
122.             p=p->link;
123.         p->link=temp;
124.         temp->link=NULL;
125.     }
126. }
127. return start;
128. }
129.
130. void display (struct stuff *start)
131. { struct stuff *p;
132.   int sum=0;
133.   int j=0;
134.
135.   if(start==NULL)
136.   {
137.       printf("\n*Queue Underflow*\n");
138.   }
```

```

139.         else
140.         {
141.             p=start;
142.             printf("\n\nS.No.\t\tstuff\t\tid\t\tQuantity\tPrice\t\tTotal Price\n\n");
143.             while(p!=NULL)
144.             {
145.                 printf("%d\t\t%s\t\t%d\t\t%d\t\t%d\t\t%d\n", ++j, p->eng_stuff, p->id, p->quantity, p-
>price, (p->price*p->quantity));
146.                 p=p->link;
147.             }
148.         }
149.         printf("\n");
150.
151.     }
152.
153.     struct stuff *insert(struct stuff *start)
154.     { int pos;
155.
156.         struct stuff *p, *temp;
157.         temp=(struct stuff*)malloc(sizeof(struct stuff)*1);
158.         printf("Enter location you want to insert:");
159.         scanf("%d",&pos);
160.
161.         if(pos==1)
162.         {
163.             printf("\nEnter Name of eng_stuff %d:", i+1);
164.             scanf("%s", (temp->eng_stuff));
165.             printf("id:");
166.             scanf("%d", &(temp->id));
167.             printf("Quantity :");
168.             scanf("%d", &(temp->quantity));
169.             printf("Price of One eng_stuff:");
170.             scanf("%d", &(temp->price));
171.             temp->link=start;
172.             start=temp;

```

```

173.
174.     return start;
175. }
176.     p=start;
177.     for(i=1;i<pos-1 && p!=NULL;i++)
178.         p=p->link;
179.     if(p==NULL)
180.         printf("\n*less no of items*\n\n");
181.
182.     else
183.     {
184.         printf("\nEnter Name of eng_stuff %d:",i+1);
185.         scanf("%s",(temp->eng_stuff));
186.         printf("id:");
187.         scanf("%d",&(temp->id));
188.         printf("Quantity :");
189.         scanf("%d",&(temp->quantity));
190.         printf("Price of One eng_stuff:");
191.         scanf("%d",&(temp->price));
192.         temp->link=p->link;
193.         p->link=temp;
194.     }
195.     return start;
196. }
197.
198. struct stuff *del(struct stuff *start,char item[])
199. {
200.
201.     struct stuff *temp,*p;
202.
203.     if(strcmp(start->eng_stuff,item)==0)
204.     {
205.         temp=start;
206.         start=temp->link;
207.         free(temp);

```

```

208.         return start;
209.     }
210.     p=start;
211.     while(p->link!=NULL)
212.     { if(strcmp(p->link->eng_stuff,item)==0)
213.         {
214.             temp=p->link;
215.             p->link=temp->link;
216.             free(temp);
217.             return start;
218.         }
219.     p=p->link;
220.     }
221. }
222.
223. struct stuff *sort (struct stuff *start)
224. {
225.     struct stuff *p,*q,*temp;
226.     temp=(struct stuff*)malloc(sizeof(struct stuff)*1);
227.
228.     for(p=start;p->link!=NULL;p=p->link)
229.     {
230.         for(q=p->link;q!=NULL;q=q->link)
231.         {
232.
233.             if(p->price > q->price)
234.             {
235.                 temp->price=p->price;
236.                 p->price=q->price;
237.                 q->price=temp->price;
238.                 strcpy(temp->eng_stuff,p->eng_stuff);
239.                 strcpy(p->eng_stuff,q->eng_stuff);
240.                 strcpy(q->eng_stuff,temp->eng_stuff);
241.                 temp->quantity=p->quantity;
242.                 p->quantity=q->quantity;

```

```

243.         q->quantity=temp->quantity;
244.         temp->id=p->id;
245.         p->id=q->id;
246.         q->id=temp->id;
247.
248.     }
249. }
250. }
251. }
252.
253. void *search(struct stuff *start,char item[])
254. {
255.     struct stuff *p;
256.     int flag=0;
257.     int j=0;
258.
259.     if(strcmp(start->eng_stuff,item)==0)
260.     {
261.         printf("\n*ITEM FOUND*\n");
262.         printf("\n\nS.No.\t\tstuff\t\tid\t\tQuantity\tPrice\t\tTotal Price\n\n");
263.         printf("%d\t\t%s\t\t%d\t\t%d\t\t%d\t\t%d\n",++j,start->eng_stuff,start->id,start-
>quantity,start->price,(start->price*p->quantity));
264.         flag=1;
265.     }
266.
267.
268.     p=start;
269.     while(p->link!=NULL)
270.     {
271.         if(strcmp(p->link->eng_stuff,item)==0)
272.         {
273.             printf("\n*ITEM FOUND*\n");
274.             printf("\n\nS.No.\t\tstuff\t\tid\t\tQuantity\tPrice\t\tTotal Price\n\n");
275.             printf("%d\t\t%s\t\t%d\t\t%d\t\t%d\t\t%d\n",++j,p->link->eng_stuff,p->link->id,p-
>link->quantity,p->link->price,p->link->price*p->link->quantity);
276.

```



```

277.         }
278.         p=p->link;
279.         flag=1;
280.     }
281.
282.     if(flag==0)
283.         printf("\n\nnot found\n\n");
284.
285. }
286.
287. struct stuff *modify(struct stuff *start,char item[])
288. {
289.     struct stuff *p;
290.     int choice;
291.     char ans1,ans2;
292.     int flag=0,chw=0;
293.     int j=0;
294.
295.     if(strcmp(start->eng_stuff,item)==0)
296.     {
297.         printf("\nITEM FOUND\n");
298.         printf("\n\nSerial no\tstuff\t\tid\t\tQuantity\tPrice\t\tTotal Price\n\n");
299.         printf("%d\t\t%s\t\t%d\t\t%d\t\t%d\t\t%d\n",++j,start->eng_stuff,start->id,start-
>quantity,start->price,start->price*start->quantity);
300.
301.     do
302.     {
303.         printf("enter:\n 1) to modify eng_stuff name\n 2) to modify eng_stuff price\n 3) to
modify eng_stuff id\n 4) to modify eng_stuff quantity:");
304.         scanf("%d",&choice);
305.         switch (choice)
306.         {
307.             case 1:
308.                 printf("enter new eng_stuff name:");
309.                 scanf("%s",(start->eng_stuff));
310.                 break;

```

```

311.         case 2:
312.             printf("enter new eng_stuff price:");
313.             scanf("%d",&(start->price));
314.             break;
315.         case 3:
316.             printf("enter new eng_stuff id:");
317.             scanf("%s",(start->id));
318.             break;
319.         case 4:
320.             printf("enter new eng_stuff quantity:");
321.             scanf("%d",&(start->quantity));
322.             break;
323.     }
324.     printf("press 1 to continue modifying:");
325.     scanf("%d",&chw);
326. }
327. while(chw==1);
328.     flag++;
329.     return start;
330. }
331. p=start;
332. while(p->link!=NULL)
333. {
334.     if(strcmp(p->link->eng_stuff,item)==0)
335.     {
336.         printf("\n*ITEM FOUND\n");
337.         printf("\n\nSerial no\tstuff\t\tid\t\tQuantity\tPrice\t\tTotal Price\n\n");
338.         printf("%d\t\t%s\t\t%d\t\t%d\t\t%d\t\t%d\n",++j,p->link->eng_stuff,p->link->id,p-
>link->quantity,p->link->price,p->link->price*p->link->quantity);
339.
340.         do
341.         {
342.             printf("enter:\n 1) to modify eng_stuff name\n 2) to modify eng_stuff price\n 3)
to modify eng_stuff id\n 4) to modify eng_stuff quantity:");
343.             scanf("%d",&choice);
344.

```

```

345.         switch (choice)
346.         {
347.             case 1:
348.                 printf("enter new eng_stuff name:");
349.                 scanf("%s", (p->link->eng_stuff));
350.                 break;
351.             case 2:
352.                 printf("enter new eng_stuff price:");
353.                 scanf("%d", &(p->link->price));
354.                 break;
355.             case 3:
356.                 printf("enter new eng_stuff id:");
357.                 scanf("%s", (p->link->id));
358.                 break;
359.             case 4:
360.                 printf("enter new eng_stuff quantity:");
361.                 scanf("%d", &(p->link->quantity));
362.                 break;
363.
364.             }printf("press 1 to continue modifying:");
365.             scanf("%d", &chw);
366.         }
367.         while(chw==1);
368.     }
369.     p=p->link;
370.     flag++;
371.     return start;
372. }
373.
374.     if(flag==0)
375.         printf("\n\nnot found\n\n");
376. }
377.
378.
379. void create_queue()

```

```

380.     {
381.         struct stuff *temp,*p;
382.         int n;
383.         printf("Enter Number of types of orders:");
384.         scanf("%d",&n);
385.         for(i=0;i<n;i++)
386.         {
387.             if (rear==NULL)
388.             {
389.                 rear=(struct stuff*)malloc(sizeof(struct stuff)*100);
390.                 printf("\nEnter Name of eng_stuff %d:",i+1);
391.                 scanf("%s",(rear->eng_stuff));
392.                 printf("id:");
393.                 scanf("%d",&(rear->id));
394.                 printf("Quantity :");
395.                 scanf("%d",&(rear->quantity));
396.                 printf("Price of One eng_stuff:");
397.                 scanf("%d",&(rear->price));
398.                 rear->link=rear;
399.                 front=rear;
400.             }
401.             else
402.             {
403.                 temp=(struct stuff*)malloc(sizeof(struct stuff));
404.                 printf("\nEnter Name of eng_stuff %d:",i+1);
405.                 scanf("%s",(temp->eng_stuff));
406.                 printf("id:");
407.                 scanf("%d",&(temp->id));
408.                 printf("Quantity :");
409.                 scanf("%d",&(temp->quantity));
410.                 printf("Price of One eng_stuff:");
411.                 scanf("%d",&(temp->price));
412.                 rear->link=temp;
413.                 temp->link=NULL;
414.                 rear=temp;

```

```

415.     }
416.     }
417. }

418. void display_queue(struct stuff *front)
419. {
420.     struct stuff *p;
421.     int sum=0;
422.     int j=0;
423.
424.     if(front==NULL)
425.     {
426.         printf("\n*Queue Underflow*\n");
427.     }
428.     else
429.     {
430.         p=front;
431.         printf("\n\nS.No.\t\tstuff\t\tid\t\tQuantity\tPrice\t\tTotal Price\n\n");
432.         while(p!=NULL)
433.         {
434.             printf("%d\t\t%s\t\t%d\t\t%d\t\t%d\t\t%d\n",++j,p->eng_stuff,p->id,p->quantity,p-
>price,(p->price*p->quantity));
435.             p=p->link;
436.         }
437.     }
438. }

439.
440. void del_queue()
441. {
442.     struct stuff *temp;
443.     if(front==NULL)
444.     {
445.         printf("\n*Queue Underflow*\n");
446.     }
447.     else

```

```

448.     {
449.         temp=front;
450.         front=front->link;
451.         free (temp);
452.     }
453. }
454. //creating a graph function
455. void graph(struct stuff *start,char item[20])
456. {
457.     int tempqty,flag;
458.     if(strcmp(start->eng_stuff,item)==0)
459.     {
460.         printf("Quantity of eng_stuff %s is %d :",start->eng_stuff,start->quantity);
461.         tempqty=start->quantity;
462.         if((tempqty)>=5)
463.         {
464.             printf("\n\n***In Profit !!!***");
465.
466.             for(i=0; i<tempqty; i++)
467.             {
468.                 printf("\n \t\t||| \n");
469.             }
470.             printf("\n \t\t%s",start->eng_stuff);
471.         }
472.
473.     else
474.     {
475.         printf("\n\n***In Loss !!!***");
476.         for(i=0; i<tempqty; i++)
477.         {
478.             printf("\n \t\t|| \n");
479.         }
480.         printf("\n \t\t%s",start->eng_stuff);
481.     }
482.     flag++;

```

```

483.     }
484.     p=start;
485.
486.     while(p->link!=NULL)
487.     {
488.         if(strcmp(p->link->eng_stuff,item)==0)
489.         {
490.             printf("Quantity of eng_stuff %s is %d :",p->link->eng_stuff,p->link->quantity);
491.             tempqty=p->link->quantity;
492.             if((tempqty)>=50)
493.             {
494.                 printf("\n\n***In Profit !!!***");
495.                 for(i=0; i<tempqty; i++)
496.                 {
497.                     printf("\n \t\t||| \n");
498.                 }
499.                 printf("\n \t\t%s",p->link->eng_stuff);
500.             }
501.             else
502.             {
503.                 printf("\n\n***In Loss !!!***");
504.                 for(i=0; i<tempqty; i++)
505.                 {
506.                     printf("\n \t\t|| \n");
507.                 }
508.                 printf("\n \t\t%s",p->link->eng_stuff);
509.             }
510.             flag++;
511.             p=p->link;
512.
513.         }
514.         if(flag==0)
515.             printf("\n\n**No item found**\n\n");
516.     }

```

output:

```
1.  #include<stdio.h>
2.  #include<stdlib.h>
3.  #include<string.h>
4.
5.  struct stuff
6.  {
7.      char eng_stuff[20];
8.      int price;
9.      int id;
10.     int quantity;
11.     struct stuff *link;
12. };
13. struct stuff *create(struct stuff *start);
14. void display (struct stuff *start);
15. struct stuff *insert(struct stuff *start);
16. struct stuff *del(struct stuff *start,char item[]);
17. struct stuff *sort (struct stuff *start);
18. void *search(struct stuff *start,char item[]);
19. struct stuff *modify(struct stuff *start,char item[]);
20. struct stuff *front=NULL;
21. struct stuff *rear=NULL;
22. void create_queue();
23. void del_queue();
24. void display_queue(struct stuff *front);
25. void graph(struct stuff *start,char [20]);
26. struct stuff *p,*q,*temp;
27. int j=0,flag=0,sum=0;
28. int i,n;
29. void main()
30. {
31.     printf("\n\t\t\t\t\t*****WELCOME TO NOBEL_OF_NETWORK ENG_STUFF*****\n");
32.     char item[20];
33.     int ch;
```



```
34. struct stuff *start=NULL;
35. while(1)
36. {
37.     printf("\nEnter choice-\n 1) to create\n 2) to create using queue\n 3) to display\n 4) to display
        using queue\n 5) to insert\n 6) to delete\n");
38.     printf(" 7) to delete using queue\n 8) to sort by price\n 9) to search\n 10) to modify:\n 11) To
        check in Profit or in Loss\t:");
39.     scanf("%d",&ch);
40.     switch (ch)
41.     {
42.     case 1:
43.         start=create(start);
44.         break;
45.     case 2:
46.         create_queue();
47.         break;
48.     case 3:
49.         display(start);
50.         break;
51.     case 4:
52.         display_queue(front);
53.         break;
54.     case 5:
55.         start=insert(start);
56.         break;
57.     case 6:
58.         printf("Enter stuff you want to delete:");
59.         scanf("%s",item);
60.         start=del(start,item);
61.         break;
62.     case 7:
63.         del_queue();
64.         break;
65.     case 8:
66.         sort(start);
67.         break;
```

```

68. case 9:
69.     printf("Enter stuff you want to search:");
70.     scanf("%s",item);
71.     search(start,item);
72.     break;
73. case 10:
74.     printf("Enter stuff you want to modify:");
75.     scanf("%s",item);
76.     start=modify(start,item);
77.     break;
78. case 11:
79.     printf("\n\t\t\t***|||*** represent PROFIT");
80.     printf("\n\t\t\t***|||*** represent LOSS");
81.     printf("\nEnter item for it's profit / loss graph:\t");
82.     scanf("%s",item);
83.     graph(start,item);
84.     break;
85. default:
86.     printf("error!");
87. }}}
88. struct stuff *create(struct stuff* start)
89. {
90.     printf("Enter no of stuff:");
91.     scanf("%d",&n);
92.     temp=start;
93.     for(i=0;i<n;i++)
94.     { temp=(struct stuff*)malloc(sizeof(struct stuff)*1);
95.         printf("\nEnter Name of eng_stuff %d:",i+1);
96.         scanf("%s",(temp->eng_stuff));
97.         printf("id of stuff:");
98.         scanf("%d",&(temp->id));
99.         printf("Quantity :");
100.             scanf("%d",&(temp->quantity));
101.             printf("Price of One eng_stuff:");
102.             scanf("%d",&(temp->price));

```

```

103.         if(start==NULL)
104.         {
105.             start=temp;
106.             temp->link=NULL;
107.         }
108.         else
109.         {
110.             p=start;
111.             while(p->link!=NULL)
112.                 p=p->link;
113.             p->link=temp;
114.             temp->link=NULL;
115.         }
116.     }
117.     return start;
118. }
119. void display (struct stuff *start)
120. { struct stuff *p;
121.   int sum=0;
122.   int j=0;
123.   if(start==NULL)
124.   {
125.       printf("\n*Queue Underflow*\n");
126.   }
127.   else
128.   {
129.       p=start;
130.       printf("\n\nS.No.\t\tstuff\t\tid\t\tQuantity\tPrice\t\tTotal Price\n\n");
131.       while(p!=NULL)
132.       {
133.           printf("%d\t\t%s\t\t%d\t\t%d\t\t%d\t\t%d\n",++j,p->eng_stuff,p->id,p->quantity,p-
>price,(p->price*p->quantity));
134.           p=p->link;
135.       }
136.   }

```

```

137.         printf("\n");
138.     }
139.     struct stuff *insert(struct stuff *start)
140.     { int pos;
141.       temp=(struct stuff*)malloc(sizeof(struct stuff)*1);
142.       printf("Enter location you want to insert:");
143.       scanf("%d",&pos);
144.       if(pos==1)
145.       {
146.         printf("\nEnter Name of eng_stuff %d:",i+1);
147.         scanf("%s",(temp->eng_stuff));
148.         printf("id:");
149.         scanf("%d",&(temp->id));
150.         printf("Quantity :");
151.         scanf("%d",&(temp->quantity));
152.         printf("Price of One eng_stuff:");
153.         scanf("%d",&(temp->price));
154.         temp->link=start;
155.         start=temp;
156.         return start;
157.     }
158.     p=start;
159.     for(i=1;i<pos-1 && p!=NULL;i++)
160.         p=p->link;
161.     if(p==NULL)
162.         printf("\n*less no of items*\n\n");
163.     else
164.     {
165.         printf("\nEnter Name of eng_stuff %d:",i+1);
166.         scanf("%s",(temp->eng_stuff));
167.         printf("id:");
168.         scanf("%d",&(temp->id));
169.         printf("Quantity :");
170.         scanf("%d",&(temp->quantity));
171.         printf("Price of One eng_stuff:");

```

```

172.         scanf("%d",&(temp->price));
173.         temp->link=p->link;
174.         p->link=temp;
175.     }
176.     return start;
177. }
178. struct stuff *del(struct stuff *start,char item[])
179. {
180.     struct stuff *temp,*p;
181.
182.     if(strcmp(start->eng_stuff,item)==0)
183.     {
184.         temp=start;
185.         start=temp->link;
186.         free(temp);
187.         return start;
188.     }
189.     p=start;
190.     while(p->link!=NULL)
191.     { if(strcmp(p->link->eng_stuff,item)==0)
192.         {
193.             temp=p->link;
194.             p->link=temp->link;
195.             free(temp);
196.             return start;
197.         }
198.         p=p->link;
199.     }
200. }
201. struct stuff *sort (struct stuff *start)
202. { struct stuff *p,*q,*temp;
203.     temp=(struct stuff*)malloc(sizeof(struct stuff)*1);
204.     for(p=start;p->link!=NULL;p=p->link)
205.     {
206.         for(q=p->link;q!=NULL;q=q->link)

```

```

207.         {
208.         if(p->price > q->price)
209.         {
210.             temp->price=p->price;
211.             p->price=q->price;
212.             q->price=temp->price;
213.             strcpy(temp->eng_stuff,p->eng_stuff);
214.             strcpy(p->eng_stuff,q->eng_stuff);
215.             strcpy(q->eng_stuff,temp->eng_stuff);
216.             temp->quantity=p->quantity;
217.             p->quantity=q->quantity;
218.             q->quantity=temp->quantity;
219.             temp->id=p->id;
220.             p->id=q->id;
221.             q->id=temp->id;
222.
223.         }}}
224. void *search(struct stuff *start,char item[])
225. {
226.     if(strcmp(start->eng_stuff,item)==0)
227.     {
228.         printf("\n*ITEM FOUND*\n");
229.         printf("\n\nS.No.\t\tstuff\t\tid\t\tQuantity\tPrice\t\tTotal Price\n\n");
230.         printf("%d\t\t%s\t\t%d\t\t%d\t\t%d\t\t%d\n",++j,start->eng_stuff,start->id,start-
>quantity,start->price,(start->price*p->quantity));
231.         flag=1;
232.     }
233.     p=start;
234.     while(p->link!=NULL)
235.     {
236.         if(strcmp(p->link->eng_stuff,item)==0)
237.         {
238.             printf("\n*ITEM FOUND*\n");
239.             printf("\n\nS.No.\t\tstuff\t\tid\t\tQuantity\tPrice\t\tTotal Price\n\n");
240.             printf("%d\t\t%s\t\t%d\t\t%d\t\t%d\t\t%d\n",++j,p->link->eng_stuff,p->link->id,p-
>link->quantity,p->link->price,p->link->price*p->link->quantity);

```

```

241.         }
242.         p=p->link;
243.         flag=1;
244.     }
245.     if(flag==0)
246.         printf("\n\nnot found\n\n");
247. }
248. struct stuff *modify(struct stuff *start,char item[])
249. {
250.     struct stuff *p;
251.     int choice;
252.     char ans1,ans2;
253.     int flag=0,chw=0;
254.     int j=0;
255.     if(strcmp(start->eng_stuff,item)==0)
256.     {
257.         printf("\nITEM FOUND\n");
258.         printf("\n\nSerial no\tstuff\t\tid\t\tQuantity\tPrice\t\tTotal Price\n\n");
259.         printf("%d\t\t%s\t\t%d\t\t%d\t\t%d\t\t%d\n",++j,start->eng_stuff,start->id,start-
>quantity,start->price,start->price*start->quantity);
260.     do
261.     {
262.         printf("enter:\n 1) to modify eng_stuff name\n 2) to modify eng_stuff price\n 3) to
        modify eng_stuff id\n 4) to modify eng_stuff quantity:");
263.         scanf("%d",&choice);
264.         switch (choice)
265.         {
266.             case 1:
267.                 printf("enter new eng_stuff name:");
268.                 scanf("%s",(start->eng_stuff));
269.                 break;
270.             case 2:
271.                 printf("enter new eng_stuff price:");
272.                 scanf("%d",&(start->price));
273.                 break;
274.             case 3:

```



```

309.         printf("enter new eng_stuff price:");
310.         scanf("%d",&(p->link->price));
311.     break;
312.     case 3:
313.         printf("enter new eng_stuff id:");
314.         scanf("%s",(p->link->id));
315.     break;
316.     case 4:
317.         printf("enter new eng_stuff quantity:");
318.         scanf("%d",&(p->link->quantity));
319.     break;
320.     }printf("press 1 to continue modifying:");
321.     scanf("%d",&chw);
322.     }
323.     while(chw==1);
324.     }
325.     p=p->link;
326.     flag++;
327.     return start;
328.     }
329.     if(flag==0)
330.         printf("\n\nnot found\n\n");
331.     }
332. void create_queue()
333. {
334.     struct stuff *temp,*p;
335.     int n;
336.     printf("Enter Number of types of orders:");
337.     scanf("%d",&n);
338.     for(i=0;i<n;i++)
339.     {
340.         if (rear==NULL)
341.         {
342.             rear=(struct stuff*)malloc(sizeof(struct stuff)*100);
343.             printf("\nEnter Name of eng_stuff %d:",i+1);

```

```

344.         scanf("%s",(rear->eng_stuff));
345.         printf("id:");
346.         scanf("%d",&(rear->id));
347.         printf("Quantity :");
348.         scanf("%d",&(rear->quantity));
349.         printf("Price of One eng_stuff:");
350.         scanf("%d",&(rear->price));
351.         rear->link=rear;
352.         front=rear;
353.     }
354.     else
355.     {
356.         temp=(struct stuff*)malloc(sizeof(struct stuff));
357.         printf("\nEnter Name of eng_stuff %d:",i+1);
358.         scanf("%s",(temp->eng_stuff));
359.         printf("id:");
360.         scanf("%d",&(temp->id));
361.         printf("Quantity :");
362.         scanf("%d",&(temp->quantity));
363.         printf("Price of One eng_stuff:");
364.         scanf("%d",&(temp->price));
365.         rear->link=temp;
366.         temp->link=NULL;
367.         rear=temp;
368.     }
369. }
370. }
371. void display_queue(struct stuff *front)
372. {
373.     struct stuff *p;
374.     int sum=0;
375.     int j=0;
376.     if(front==NULL)
377.     {
378.         printf("\n*Queue Underflow*\n");

```

```

379.         }
380.         else
381.         {
382.             p=front;
383.             printf("\n\nS.No.\t\tstuff\t\tid\t\tQuantity\tPrice\t\tTotal Price\n\n");
384.             while(p!=NULL)
385.             {
386.                 printf("%d\t\t%s\t\t%d\t\t%d\t\t%d\t\t%d\n",++j,p->eng_stuff,p->id,p->quantity,p-
>price,(p->price*p->quantity));
387.                 p=p->link;
388.             }
389.         }
390.     }
391.     void del_queue()
392.     {
393.         struct stuff *temp;
394.         if(front==NULL)
395.         {
396.             printf("\n*Queue Underflow*\n");
397.         }
398.         else
399.         {
400.             temp=front;
401.             front=front->link;
402.             free (temp);
403.         }
404.     }
405.     void graph(struct stuff *start,char item[20])
406.     {int tempqty,flag;
407.         if(strcmp(start->eng_stuff,item)==0)
408.         {
409.             printf("Quantity of eng_stuff %s is %d :",start->eng_stuff,start->quantity);
410.             tempqty=start->quantity;
411.             if((tempqty)>=5)
412.             {

```

```

413.         printf("\n\n***In Profit !!!***");
414.         for(i=0; i<tempqty; i++)
415.         {
416.             printf("\n \t\t||| \n");
417.         }
418.         printf("\n \t\t%s",start->eng_stuff);
419.     }
420. else
421. {
422.     printf("\n\n***In Loss !!!***");
423.     for(i=0; i<tempqty; i++)
424.     {
425.         printf("\n \t\t|| \n");
426.     }
427.     printf("\n \t\t%s",start->eng_stuff);
428. }
429. flag++;
430. }
431. p=start;
432. while(p->link!=NULL)
433. {
434.     if(strcmp(p->link->eng_stuff,item)==0)
435.     {
436.         printf("Quantity of eng_stuff %s is %d :",p->link->eng_stuff,p->link->quantity);
437.         tempqty=p->link->quantity;
438.         if((tempqty)>=5)
439.         {
440.             printf("\n\n***In Profit !!!***");
441.             for(i=0; i<tempqty; i++)
442.             {
443.                 printf("\n \t\t||| \n");
444.             }
445.             printf("\n \t\t%s",p->link->eng_stuff);
446.         }
447.     }

```

```
448.         {
449.             printf("\n\n***In Loss !!!***");
450.             for(i=0; i<tempqty; i++)
451.             {
452.                 printf("\n \t\t|| \n");
453.             }
454.             printf("\n \t\t%s",p->link->eng_stuff);
455.         }
456.         flag++;
457.     } p=p->link;
458. }
459. if(flag==0)
460.     printf("\n\n**No item found**\n\n");
461. }
```

input line no=516

optimise line no=461