Assignment 7

Code:-

```
#include<stdio.h>
#include<stdlib.h>
#include<string.h>

struct stuff
{
   char product[20];
   int price;
   int  id;
   int quantity;

   struct stuff *link;
};

struct stuff *create(struct stuff *start);
void display (struct stuff *start);
struct stuff *insert(struct stuff *start);
struct stuff *del(struct stuff *start,char item[]);
struct stuff *sort (struct stuff *start);
void *search(struct stuff *start,char item[]);
struct stuff *modify(struct stuff *start,char item[]);
struct stuff *front=NULL;
struct stuff *rear=NULL;
void create_queue();
void del_queue();
void display_queue(struct stuff *front);

int i,n;
void main()
```

```
{
   printf("\n\t\t\t\t******WELCOME TO NOBEL_OF_NETWORK ENG_STUFF******\n");
   char item[20];
   int ch;
   struct stuff *start=NULL;
   while(1)
   {
   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");
   printf(" 7) to delete using queue\n 8) to sort by price\n 9) to search\n 10) to modify: ");
   scanf("%d",&ch);
   switch (ch)
   {
   case 1:
      start=create(start);
      break;
   case 2:
      create_queue();
      break;
   case 3:
      display(start);
      break;
   case 4:
      display_queue(front);
      break;
   case 5:
      start=insert(start);
      break;
   case 6:
      printf("Enter item you want to delete:");
      scanf("%s",item);
```

```
        start=del(start,item);
        break;
    case 7:
        del_queue();
        break;
    case 8:
        sort(start);
        break;
    case 9:
        printf("Enter item you want to search:");
        scanf("%s",item);
        search(start,item);
        break;
    case 10:
        printf("Enter item you want to modify:");
        scanf("%s",item);
        start=modify(start,item);
        break;
    default:
        printf("error!");

    }
}
}
struct stuff *create(struct stuff* start)
{
    struct stuff *p,*temp;
    printf("Enter no of type of items:");
    scanf("%d",&n);

    temp=start;
```

```
   for(i=0;i<n;i++)
    { temp=(struct stuff*)malloc(sizeof(struct stuff)*1);
      printf("\nEnter Name of Product %d:",i+1);
      scanf("%s",(temp->product));
      printf("id of stuff:");
      scanf("%d",(temp->id));
      printf("Quantity :");
      scanf("%d",&(temp->quantity));
      printf("Price of One Product:");
      scanf("%d",&(temp->price));

   if(start==NULL)
   {
    start=temp;
    temp->link=NULL;
   }
  else
  {
       p=start;
       while(p->link!=NULL)
       p=p->link;
       p->link=temp;
       temp->link=NULL;
  }
}
return start;
}

void display (struct stuff *start)
{ struct stuff *p;
  int sum=0;
```

```
 int j=0;


  if(start==NULL)
  {
     printf("\n*Queue Underflow*\n");
  }
  else
  {
     p=start;
     printf("\n\nS.No.\t\tstuff\t\tid\t\tQuantity\tPrice\t\tTotal Price\n\n");
     while(p!=NULL)
     {
         printf("%d\t\t%s\t\t%d\t\t%d\t\t%d\t\t%d\n",++j,p->product,p->id,p->quantity,p->price,(p-
>price*p->quantity));
         p=p->link;
     }
  }
    printf("\n");


}


struct stuff *insert(struct stuff *start)
{ int pos;


 struct stuff *p,*temp;
 temp=(struct stuff*)malloc(sizeof(struct stuff)*1);
 printf("Enter location you want to insert:");
 scanf("%d",&pos);


  if(pos==1)
 {
```

```
    printf("\nEnter Name of Product %d:",i+1);
    scanf("%s",(temp->product));
    printf("id:");
    scanf("%d",&(temp->id));
    printf("Quantity :");
    scanf("%d",&(temp->quantity));
    printf("Price of One Product:");
    scanf("%d",&(temp->price));
    temp->link=start;
    start=temp;

  return start;
}
   p=start;
   for(i=1;i<pos-1 && p!=NULL;i++)
      p=p->link;
   if(p==NULL)
      printf("\n*less no of items*\n\n");

  else
  {
    printf("\nEnter Name of Product %d:",i+1);
    scanf("%s",(temp->product));
    printf("id(small/medium/large):");
    scanf("%d",&(temp->id));
    printf("Quantity :");
    scanf("%d",&(temp->quantity));
    printf("Price of One Product:");
    scanf("%d",&(temp->price));
    temp->link=p->link;
    p->link=temp;
```

```
   }
  return start;
}

struct stuff *del(struct stuff *start,char item[])
{
 struct stuff *temp,*p;

 if(strcmp(start->product,item)==0)
 {
    temp=start;
    start=temp->link;
    free(temp);
    return start;
 }
    p=start;
    while(p->link!=NULL)
    { if(strcmp(p->link->product,item)==0)
       {
         temp=p->link;
         p->link=temp->link;
         free(temp);
         return start;
    }
   p=p->link;
    }
    }

struct stuff *sort (struct stuff *start)
{   struct stuff *p,*q,*temp;
   temp=(struct stuff*)malloc(sizeof(struct stuff)*1);
```

```
  for(p=start;p->link!=NULL;p=p->link)
   {
      for(q=p->link;q!=NULL;q=q->link)
      {

      if(p->price > q->price)
       {
          temp->price=p->price;
          p->price=q->price;
          q->price=temp->price;
          strcpy(temp->product,p->product);
          strcpy(p->product,q->product);
          strcpy(q->product,temp->product);
          temp->quantity=p->quantity;
          p->quantity=q->quantity;
          q->quantity=temp->quantity;
          strcpy(temp->id,p->id);
          strcpy(p->id,q->id);
          strcpy(q->id,temp->id);

       }
     }
   }
}

void *search(struct stuff *start,char item[])
{
  struct stuff *p;
  int flag=0;
  int j=0;
```

```
    if(strcmp(start->product,item)==0)
      {
       printf("\n*ITEM FOUND*\n");
       printf("\n\nS.No.\t\tstuff\t\tid\t\tQuantity\tPrice\t\tTotal Price\n\n");
       printf("%d\t\t%s\t\t%d\t\t%d\t\t%d\t\t%d\n",++j,start->product,start->id,start->quantity,start->price,(start->price*p->quantity));
        flag=1;
      }

      p=start;
      while(p->link!=NULL)
       {
       if(strcmp(p->link->product,item)==0)
       {
        printf("\n*ITEM FOUND*\n");
         printf("\n\nS.No.\t\tstuff\t\tid\t\tQuantity\tPrice\t\tTotal Price\n\n");
       printf("%d\t\t%s\t\t%d\t\t%d\t\t%d\t\t%d\n",++j,p->link->product,p->link->id,p->link->quantity,p->link->price,p->link->price*p->link->quantity);

        }
        p=p->link;
        flag=1;
      }

      if(flag==0)
      printf("\n\nnot found\n\n");

}

struct stuff *modify(struct stuff *start,char item[])
```

```
{
   struct stuff *p;
   int choice;
   char ans1,ans2;
   int flag=0,chw=0;
   int j=0;


    if(strcmp(start->product,item)==0)
    {
     printf("\nITEM FOUND\n");
     printf("\n\nSerial no\tstuff\t\tid\t\tQuantity\tPrice\t\tTotal Price\n\n");
     printf("%d\t\t%s\t\t%d\t\t%d\t\t%d\t\t%d\n",++j,start->product,start->id,start->quantity,start->price,start->price*start->quantity);


   do
    {
     printf("enter:\n 1) to modify Product name\n 2) to modify Product price\n 3) to modify product id\n 4) to modify product quantity:");
     scanf("%d",&choice);
     switch (choice)
     {
     case 1:
      printf("enter new product name:");
      scanf("%s",(start->product));
      break;
     case 2:
      printf("enter new product price:");
      scanf("%d",&(start->price));
      break;
     case 3:
      printf("enter new product id:");
      scanf("%s",(start->id));
```

```
      break;
     case 4:
      printf("enter new product quantity:");
      scanf("%d",&(start->quantity));
      break;
     }
     printf("press 1 to continue modifying:");
     scanf("%d",&chw);
   }
   while(chw==1);
        flag++;
        return start;
   }
    p=start;
    while(p->link!=NULL)
     {
       if(strcmp(p->link->product,item)==0)
      {
       printf("\n*ITEM FOUND\n");
       printf("\n\nSerial no\tstuff\t\tid\t\tQuantity\tPrice\t\tTotal Price\n\n");
       printf("%d\t\t%s\t\t%d\t\t%d\t\t%d\t\t%d\n",++j,p->link->product,p->link->id,p->link->quantity,p->link->price,p->link->price*p->link->quantity);

        do
        {
         printf("enter:\n 1) to modify Product name\n 2) to modify Product price\n  3) to modify product id\n 4) to modify product quantity:");
         scanf("%d",&choice);

       switch (choice)
       {
        case 1:
```

```
          printf("enter new product name:");
          scanf("%s",(p->link->product));
        break;
        case 2:
          printf("enter new product price:");
          scanf("%d",&(p->link->price));
        break;
        case 3:
          printf("enter new product id:");
          scanf("%s",(p->link->id));
        break;
        case 4:
          printf("enter new product quantity:");
          scanf("%d",&(p->link->quantity));
        break;

        }printf("press 1 to continue modifying:");
         scanf("%d",&chw);
         }
         while(chw==1);
        }
           p=p->link;
           flag++;
           return start;
      }

      if(flag==0)
      printf("\n\nnot found\n\n");
}
```

```
void create_queue()
{
    struct stuff *temp,*p;
    int n;
    printf("Enter Number of types of orders:");
    scanf("%d",&n);
    for(i=0;i<n;i++)
    {
     if (rear==NULL)
   {
      rear=(struct stuff*)malloc(sizeof(struct stuff)*100);
      printf("\nEnter Name of Product %d:",i+1);
      scanf("%s",(rear->product));
      printf("id:");
      scanf("%d",&(rear->id));
      printf("Quantity :");
      scanf("%d",&(rear->quantity));
      printf("Price of One Product:");
      scanf("%d",&(rear->price));
      rear->link=rear;
      front=rear;
   }
   else
   {
      temp=(struct stuff*)malloc(sizeof(struct stuff));
      printf("\nEnter Name of Product %d:",i+1);
      scanf("%s",(temp->product));
      printf("id:");
      scanf("%d",&(temp->id));
      printf("Quantity :");
      scanf("%d",&(temp->quantity));
```

```
        printf("Price of One Product:");
        scanf("%d",&(temp->price));
        rear->link=temp;
        temp->link=NULL;
        rear=temp;
    }
     }
}

void display_queue(struct stuff *front)
{
    struct stuff *p;
    int sum=0;
    int j=0;

    if(front==NULL)
    {
        printf("\n*Queue Underflow*\n");
    }
    else
    {
        p=front;
        printf("\n\nS.No.\t\tstuff\t\tid\t\tQuantity\tPrice\t\tTotal Price\n\n");
        while(p!=NULL)
        {
            printf("%d\t\t%s\t\t%d\t\t%d\t\t%d\t\t%d\n",++j,p->product,p->id,p->quantity,p->price,(p-
>price*p->quantity));
            p=p->link;
        }
    }
}
```

```
void del_queue()
{
   struct stuff *temp;
   if(front==NULL)
   {
      printf("\n*Queue Underflow*\n");
   }
   else
   {
   temp=front;
   front=front->link;
   free (temp);
   }
}
```

OUTPUT:-

******WELCOME TO NOBEL_OF_NETWORK ENG_STUFF******

Enter choice-

1) to create
2) to create using queue
3) to display
4) to display using queue
5) to insert
6) to delete
7) to delete using queue
8) to sort by price
9) to search

10) to modify: 2

Enter Number of types of orders:2

Enter Name of Product 1:computer

id:1234

Quantity :12

Price of One Product:3200

Enter Name of Product 2:keyboard

id:1232

Quantity :34

Price of One Product:3400

Enter choice-

1) to create

2) to create using queue

3) to display

4) to display using queue

5) to insert

6) to delete

7) to delete using queue

8) to sort by price

9) to search

10) to modify: 4

S.No.	stuff	id	Quantity	Price	Total Price
1	computer	1234	12	3200	38400
2	keyboard	1232	34	3400	115600

Enter choice-

1) to create

2) to create using queue

3) to display

4) to display using queue

5) to insert

6) to delete

7) to delete using queue

8) to sort by price

9) to search

10) to modify: 7

Enter choice-

1) to create

2) to create using queue

3) to display

4) to display using queue

5) to insert

6) to delete

7) to delete using queue

8) to sort by price

9) to search

10) to modify: 4

S.No.	stuff	id	Quantity	Price	Total Price
1	keyboard	1232	34	3400	115600

Enter choice-

1) to create

2) to create using queue

3) to display

4) to display using queue

5) to insert

6) to delete

7) to delete using queue

8) to sort by price

9) to search

10) to modify: 5

Enter location you want to insert:2

less no of items

Enter choice-

1) to create

2) to create using queue

3) to display

4) to display using queue

5) to insert

6) to delete

7) to delete using queue

8) to sort by price

9) to search

10) to modify: