```
//Header Linked List

# include <stdio.h>
# include <conio.h>

struct product
{
 int code;
 char name[10];
 struct product *next;
};

struct product *start;

void main()
{
 int choice;

 void insert_first();
 void insert_last();
 void insert_specific();
 void delete_first();
 void delete_last();
 void delete_specific_nodeno();
 void delete_specific_nodevalue();
 void display();
 void search();
 void sort();

 start = (struct product *) malloc(sizeof(struct product));
 start->code = 0;
 start->next = NULL;

 do
 {
  clrscr();

  printf("\n\t1. Insert First");
  printf("\n\t2. Insert Last");
  printf("\n\t3. Insert Specific");
  printf("\n\t4. Delete First");
  printf("\n\t5. Delete Last");
  printf("\n\t6. Delete Specific by node no ");
  printf("\n\t7. Delete Specific by node value");
  printf("\n\t8. Display");
  printf("\n\t9. Search");
  printf("\n\t10. Sort");
  printf("\n\t0. Exit");

  printf("\n\tEnter your choice : ");
  scanf("%d",&choice);

  switch(choice)
  {
   case 1:
    insert_first();
    break;
   case 2:
    insert_last();
    break;
   case 3:
    insert_specific();
    break;
   case 4:
    delete_first();
    break;
   case 5:
    delete_last();
    break;
   case 6:
    delete_specific_nodeno();
    break;
   case 7:
    delete_specific_nodevalue();
    break;
   case 8:
    display();
    break;
   case 9:
    search();
    break;
   case 10:
    sort();
    break;
   case 0:
    printf("\n\tEnd of program");
    break;
   default:
    printf("\n\tInvalid Choice");
    break;
  }
  getch();
 }
 while(choice != 0);
}

void insert_first()
{
 struct product *newnode;

 newnode=(struct product *) malloc(sizeof(struct product));

 printf("\n\tEnter Product Code : ");
 scanf("%d",&newnode->code);

 printf("\n\tEnter Product Name : ");
 fflush(stdin);
 gets(newnode->name);

 newnode->next = start->next;
 start->next = newnode;

 start->code = start->code + 1;
}

void display()
{
 struct product *temp;

 if(start->next == NULL)
 {
  printf("\n\tHeader Linked List is Empty");
 }
 else
 {
  temp = start->next;
  printf("\n\tProduct Code\tProduct Name");
  while(temp != NULL)
  {
        printf("\n\t%d\t\t%s", temp->code, temp->name);
        temp = temp->next;
  }

printf("\n\n\tThere are %d nodes in header linked list",start->code);

 }
}

void insert_last()
{
 struct product *temp, *newnode;

 newnode = (struct product *) malloc(sizeof(struct product));

 printf("\n\tEnter Product Code : ");
 scanf("%d",&newnode->code);

 printf("\n\tEnter Product Name : ");
 fflush(stdin);
 gets(newnode->name);

 if(start->next  == NULL)
 {
  start->next = newnode;
  newnode->next = NULL;
 }
 else
 {
  temp = start;

  while(temp->next != NULL)
  {
   temp = temp->next;
  }
  temp->next = newnode;
  newnode->next = NULL;
 }
 start->code = start->code + 1;
}

void delete_first()
{
 struct product *delnode;

 if(start->next == NULL)
 {
  printf("\n\tHeader Linked List is Empty");
 }
 else
 {
  delnode = start->next;
  start->next = start->next->next;

  printf("\n\tDelete Node Information : ");
  printf("\n\tProduct Code : %d", delnode->code);
  printf("\n\tProduct Name : %s", delnode->name);

  free(delnode);

  start->code = start->code - 1;
 }
}

void delete_last()
{
 struct product *temp, *delnode;

 if(start->next == NULL)
 {
  printf("\n\tHeader Linked List is Empty");
 }
 else
 {
  if(start->next->next == NULL)
  {
   delnode = start->next->next;
   start->next = start->next->next;
  }
  else
  {
   temp = start->next;

   while(temp->next->next != NULL)
   {
    temp = temp->next;
   }

   delnode = temp->next;
   temp->next = NULL;
  }

  printf("\n\tDelete Node Information : ");
  printf("\n\tProduct Code : %d",delnode->code);
  printf("\n\tProduct Name : %s", delnode->name);

  free(delnode);

  start->code = start->code - 1;
 }
}

void insert_specific()
{
 struct product *newnode, *temp;
 int a, nodeno, count=0;


 if(start->next == NULL)
 {
    newnode = (struct product *) malloc(sizeof(struct product));
  start->next = newnode;
  newnode->next = NULL;

  printf("\n\tEnter Product Code : ");
  scanf("%d",&newnode->code);

  printf("\n\tEnter Product Name : ");
  fflush(stdin);
  gets(newnode->name);

  start->code = 1;
 }
 else
 {
  temp = start->next;

  while(temp != NULL)
  {
   count ++;
   temp = temp->next;
  }

  do
  {

printf("\n\tEnter Node no. to insert between 1 to %d : ", count+1);
   scanf("%d",&nodeno);
  }
  while(nodeno < 1 || nodeno > count+1);

  if(nodeno == 1)
  {
   insert_first();
  }
  else if(nodeno == count+1)
  {
   insert_last();
  }
  else
  {
   temp = start->next;
   for(a=1;a<nodeno-1;a++)
   {
    temp = temp->next;
   }

 newnode = (struct product *) malloc(sizeof(struct product));

   newnode->next = temp->next;
   temp->next = newnode;

   printf("\n\tEnter Product Code : ");
   scanf("%d",&newnode->code);

   printf("\n\tEnter Product Name : ");
   fflush(stdin);
   gets(newnode->name);

   start->code = start->code + 1;
  }
 }
}

void search()
{
 struct product *temp;
 int sv;

 if(start->next == NULL)
 {
  printf("\n\tHeader Linked List is Empty");
 }
 else
 {
  printf("\n\tEnter Product code to search : ");
  scanf("%d",&sv);

  temp = start->next;

  while(temp != NULL)
  {
   if(temp->code == sv)
   {
      printf("\n\tProduct Name : %s", temp->name);
      break;
   }

   temp = temp->next;
  }
  if(temp == NULL)
  {
   printf("\n\tProduct Code does not exists");
  }
 }
}

void sort()
{
 struct product *temp1, *temp2;
 int cd;
 char nm[10];

 if(start->next == NULL)
 {
  printf("\n\tHeader Linked List is Empty");
 }
 else
 {
  temp1 = start->next;

  while(temp1->next != NULL)
  {
   temp2 = temp1->next;

   while(temp2 != NULL)
   {
    if(temp1->code > temp2->code)
    {
     cd = temp1->code;
     temp1->code = temp2->code;
     temp2->code = cd;

     strcpy(nm, temp1->name);
        strcpy(temp1->name, temp2->name);
     strcpy(temp2->name, nm);
    }
    temp2 = temp2->next;
   }
   temp1 = temp1->next;
  }

  display();
 }
}

void delete_specific_nodeno()
{
 struct product *temp, *delnode;
 int a, nodeno, count=0;

 if(start->next == NULL)
 {
  printf("\n\tHeader Linked List is Empty");
 }
 else
 {
  temp = start->next;

  while(temp != NULL)
  {
   count++;
   temp = temp->next;
  }

  do
  {

printf("\n\tEnter node no to delete between 1 to %d : " , count);
   scanf("%d",&nodeno);
  }
  while(nodeno < 1 || nodeno > count);

  if(nodeno == 1)
  {
   delete_first();
  }
  else if(nodeno == count)
  {
   delete_last();
  }
  else
  {
   temp = start->next;

   for(a=1;a<nodeno-1;a++)
   {
    temp = temp->next;
   }

   delnode = temp->next;
   temp->next = temp->next->next;

   printf("\n\tDelete Node Information : ");
   printf("\n\tProduct Code : %d", delnode->code);
   printf("\n\tProduct Name : %s", delnode->name);

   free(delnode);

   start->code = start->code - 1;
  }
 }
}

void delete_specific_nodevalue()
{
 int sv;
 struct product *temp, *delnode = NULL;
 if(start->next == NULL)
 {
  printf("\n\tHeader Linked List is Empty");
 }
 else
 {
  printf("\n\tEnter Product code to Delete : ");
  scanf("%d",&sv);

  temp = start->next;

  if(temp->code == sv)
  {
   delete_first();
  }
  else
  {
   while(temp->next != NULL)
   {
    if(temp->next->code == sv)
    {
     delnode = temp->next;
     temp->next = temp->next->next;
     break;
    }
    temp = temp->next;
   }

   if(delnode == NULL)
   {
     printf("\n\tDelete Product code not found");
   }
   else
   {
     printf("\n\tDelete Node Information : ");
     printf("\n\tProduct Code : %d", delnode->code);
     printf("\n\tProduct Name : %s", delnode->name);

     free(delnode);

     start->code = start->code - 1;
   }
  }
 }
}
```