Answer on Question #43973, Engineering, SolidWorks | CosmoWorks | Ansys | for completion

Solve by linked list.

```
#include <stdio.h>
#include <conio.h>
#include <stdlib.h>
#include <malloc.h>
struct linked_list
{
int number;
struct linked_list *next;
};
linked_list *node;
void create (linked_list *p);
int countNode (linked_list *p);
void count();
void print (linked_list *p);
linked_list *deleteItem(linked_list *p);
linked_list *insert(linked_list *p);
int main ()
{
int n, choice;
node = (linked_list *)malloc (sizeof (linked_list));
create (node);
do {
printf("\n================");
printf("\n List operation");
printf("\n================");
printf("\n 1. Insert a New Node");
printf("\n 2. Delete existing node");
printf("\n 3. Display the item within the list");
printf("\n 4. Count the number of items");
```

```
printf("\n 5. Exit");
printf("\n what to do? press the number between 1-5: \n");
scanf("%d",&choice);
switch (choice)
{
case 1:
insert(node);
break;
case 2:
deleteItem(node);
break;
case 3:
print(node);
break;
case 4:
count();
break;
case 5:
exit(1);
break;
default:
printf("\n Invalid choice! so please select 1-5: \n");

}

} while (choice!=5);

}
void create (linked_list *list)
{
printf("\n Input the list items: (type -999 at end): ");
scanf("%d",&list -> number);
if (list -> number== -999)
{
list->next=NULL;
}
else
```

```
{
list->next=(linked_list *)malloc(sizeof (linked_list));
create (list -> next);
}
return;
}
void print (linked_list *list)
{
if (list -> next != NULL)
{

printf("%d -->",list -> number);


if (list -> next -> next == NULL)

printf("%d",list -> next -> number);


print(list -> next);
}
return;
}
int countNode (linked_list *list)
{
if (list -> next == NULL)
return 0;
else
return (1+countNode(list -> next));
}
void count ()
{
printf("\n ========================================");
printf("\n The number of items in the list is %d\n",countNode(node));
printf(" ========================================");
}
linked_list *insert (linked_list *list)
```

```
{
linked_list *find(linked_list *p, int a);
linked_list *newNode;
linked_list *nextNode;
int key; //position of where new item should be inserted
int x; /*new item (number) to be inserted */
printf ("\n Enter the value of new item?: \n");
scanf("%d",&x);
printf("\n What is the position of new item? (type -999 if last):\n");
scanf("%d",&key);
if (list -> number == key) /*new node is the first */
{
newNode = (linked_list *)malloc(sizeof(linked_list));
newNode -> number = x;
newNode -> next = list;
list = newNode;
}
else
{
nextNode = find (list,key);
if (nextNode == NULL)
{
printf("\n key is not found");
}
else
{
newNode = (linked_list *)malloc (sizeof (linked_list));
newNode -> number = x;
newNode -> next = nextNode -> next;
nextNode -> next = newNode;
}
}

print(list);

return(list);
}
```

```
linked_list *find (linked_list *list, int key)
{
if (list -> next -> number == key)
return (list);
else
{
if (list -> next -> next == NULL)
return (NULL);
else
find (list -> next,key);
}
}

linked_list *deleteItem(linked_list *list)
{
linked_list *find (linked_list *p, int a);
int key;
linked_list *nodePointer;
linked_list *tempPointer;
printf("\n What is the item to be deleted?: \n");
scanf("%d",&key);
if (list -> number == key)
{
tempPointer = list -> next;
free (list);
list = tempPointer;
}
else
{
nodePointer = find (list,key);
if (nodePointer == NULL)
printf("\n Key not found");
else
{
tempPointer = nodePointer ->next -> next;
free (nodePointer -> next);
```

```
nodePointer -> next = tempPointer;
}
}

print(list);

return(list);
}
```