Question3.cpp

```cpp
// <----Lab 03 - Singly Linked List---->

// Q3. Use the class of SLL created by you during the lab task 1. Do the following:
// a) Reverse the linked list
// b) Sort the contents of linked list
// c) Find the duplicates in the linked list

#include<iostream>
using namespace std;

class node {
    public:
        int data;
        node* next;
        node(int value) {
            data=value;
            next=NULL;
        }
        node(int value,node* nxt) {
            data=value;
            next=nxt;
        }
};


class SLL {  //SLL = Singly Linked List made in task 1
        node* head=NULL;
    public:
        void insertAttail(int value) {
            node* n=new node(value);
            if(head==NULL) {  //Check if Linked List Empty.
                head=n;
                return;
            }
            node* temp=head;
            while(temp->next!=NULL) {
                temp=temp->next;
            }
            temp->next=n;
        }
        void insertAtPos(int posvalue,int value) {
            int count=0;
            node* temp=head;
            while(temp->next!=NULL&&count<posvalue-1) {
                temp=temp->next;
                count++;
            }
            node* n=new node(value,temp->next);
            temp->next=n;
        }
        void display() {
            node* temp=head;
            cout<<"[HEAD] ";
```

```
54              while(temp!=NULL) {
55                  cout<<temp->data<<"| "<<temp->next<<" -> ";
56                  temp=temp->next;
57              }
58              cout<<"NULL [TAIL]"<<endl;
59          }
60          void insertAthead(int value) {
61              node* n=new node(value);
62              n->next=head;
63              head=n;
64          }
65          void deletion(int value) {
66              if(head==NULL) {
67                  return;
68              }
69              node* temp=head;
70              while(temp->next->data!=value ) {
71                  temp=temp->next;
72              }
73              node* todelete=temp->next;
74              temp->next=temp->next->next;
75
76              delete todelete;
77          }
78          void deleteAthead() {
79              if(head==NULL) {
80                  return;
81              }
82              node* todelete=head;
83              head=head->next;
84              delete todelete;
85          }
86          void deleteAtPos(int posvalue) {
87              if(head==NULL) {
88                  return;
89              }
90              int count=0;
91              node* temp=head;
92              while(temp->next!=NULL && count<posvalue-1) {
93                  temp=temp->next;
94              }
95              node* todelete=temp->next;
96              temp->next=temp->next->next;
97
98              delete todelete;
99
100         }
101         void deleteAttail() {
102             if(head==NULL) { //If linked list empty.
103                 return;
104             }
105             node* temp=head;
106             if(head->next==NULL) { //If linked list has 1 item only.
107                 head=NULL;
108                 delete temp;
109             }
```

```
110             while(temp->next->next!=NULL) {
111                 temp=temp->next;
112             }
113             delete temp->next;
114             temp->next=NULL;
115         }
116 
117         void sortlist() {  //sorting by swapping values
118             node* temp=head;
119             while(temp->next!=NULL) {
120                 node* temp2=head;
121                 while(temp2->next!=NULL) {
122                     if(temp2->data>temp2->next->data) {
123                         char tempchar = temp2->data;
124                         temp2->data = temp2->next->data;
125                         temp2->next->data = tempchar;
126                     }
127                     temp2=temp2->next;
128                 }
129                 temp=temp->next;
130             }
131         }
132 
133         void reverse() {
134             node* prev=NULL;
135             node* after=NULL;
136             while(head!=NULL) {
137                 after=head->next;
138                 head->next=prev;
139                 prev=head;
140                 head=after;
141             }
142             head=prev;
143         }
144 
145         void duplicates() {
146             int* items = new int[100];      // Assuming a maximum of 100 unique items
147             int* itemcount = new int[100];
148             int size = 0;
149 
150             node* temp = head;
151 
152             while(temp != NULL) {
153                 bool duplicate = false;
154                 for(int i = 0; i < size; i++) {
155                     if(temp->data == items[i]) {
156                         duplicate = true;
157                         itemcount[i]++;
158                         break;
159                     }
160                 }
161                 if(!duplicate) {
162                     items[size] = temp->data;
163                     itemcount[size] = 1;
164                     size++;
165                 }
```

```
166                 temp = temp->next;
167             }
168             cout<<"Unique Items: "<<size<<endl;
169             for(int i = 0; i < size; i++) {
170                 cout<<"Data: "<<items[i]<<", Count :"<<itemcount[i]<<endl;
171             }
172 
173             delete[] items;
174             delete[] itemcount;
175 
176         }
177 
178 };
179 
180 int main() {
181     SLL list;
182     float input=0;
183     while(input!=0.5) {
184         cout<<"Enter integer to put in linked list [Enter 0.5 to end input phase]: ";
185         cin>>input;
186         if(input!=0.5) {
187             list.insertAthead((int)input);
188         }
189     }
190     cout<<"\n\n";
191     list.display();
192     char option;
193     cout<<"\n\n\nDo you want to...  \n'R'\tReverse the Linked List.\n'S'\tSort the Linked
    List.\n'D'\tFind Duplicates in Linked List?\n";
194     cin>>option;
195     if(option=='R'||option=='r'){
196         cout<<"Reversing the Linked List."<<endl;
197         list.sortlist();
198         list.display();
199     }
200     else if(option=='S'||option=='s'){
201         cout<<"Sorting the Linked List."<<endl;
202         list.sortlist();
203         list.display();
204     }
205     else if(option=='D'||option=='d'){
206         cout<<"Finding Duplicates: "<<endl;
207         list.duplicates();
208     }
209     else{
210         cout<<"Invalid Input. defaulting to Sort."<<endl;
211         list.sortlist();
212         list.display();
213     }
214 }
```