

Name: M_uhammad S_herjeel A_khtar

R_oll N_o: 20p-0101

H_ork N_o: 5

S_ubject: D_ata S_tructures

S_ection: B_{CS}-4_B

S_ubmitted T_o R_espected S_ir: **Khurram Shahzad**

Activities Visual Studio Code رج 10:28 21 ش

main.cpp - Downloads - Visu

File Edit Selection View Go Run Terminal Help

EXPLORER

... largest.asm main.cpp x

Lab_5_Home_Wokr > main.cpp > LinkedList > insertionatend(int)

```
1 #include<iostream>
2 using namespace std;
3 class Node{
4     public:
5     int data;
6     Node* next;
7     Node(int value){
8         data = value;};
9 class LinkedList{
10     Node* head;
11     int length;
12     LinkedList(){
13         head = NULL;
14         length = 0;};
15 private:
```

```
15 private:
16 void insertionatstart(int val){
17     Node *n=new Node(val);
18     if(head==NULL){
19         head=n;};
20     else if(head!=NULL){
21         n->next = head;
22         head=n;}}
23 void insertionatend(int val){
24     Node *temp;
25     Node* n=new Node(val);
26     temp=head;
27     while(temp!=NULL){
28         temp=temp->next;};
29     temp->next=n;};
30 void insertionatmid(int val, int pos){
31     Node* n=new Node(val);
32     Node* temp;
33     temp=head;
34     for(int i=1; i<pos; i++){
35         temp=temp->next;};
```

```

35     temp=temp->next;
36     n->next=temp->next;
37     temp->next=n;
38 }
39 void clearLL(){
40     Node* temp, *prev;
41     temp=head;
42     prev=head;
43     if(head!=NULL){
44         while(temp!=NULL){
45             temp->next;
46             delete prev;
47             prev=temp;}}
48     else{
49         cout<<"There is nothing to delete in the list";}}
50 bool search(int val){
51     bool flag=false;
52     Node* temp;
53     temp=head;
54     while(temp->data!=val){
55         // temp=temp->next;
56         if(temp->data==val){
57             flag=true;
58             return flag;}
59         temp=temp->next;}
60 }

```

```

60 }
61 void printed(){
62     Node* temp;
63     temp=head;
64     while(temp!=NULL){
65         cout<<temp->data;
66         temp=temp->next;}
67 }
68 void concatenation(Node *Flist,Node *Slist){
69     bool flag=false;
70     if( Flist != NULL && Slist!= NULL ){
71         if (Flist->next == NULL)
72             Flist->next = Slist;
73         else
74             concatenation(Flist->next,Slist);
75         flag=true;}
76     else{
77         if(flag==false){
78             cout<<"The given two input lists are empty.";
79         }}}
80 bool emptiness(){
81     bool flag=false;
82     if(head==NULL){
83         flag=true;
84         return flag;
85     }
86 }

```

```

60 }
61 void printed(){
62     Node* temp;
63     temp=head;
64     while(temp!=NULL){
65         cout<<temp->data;
66         temp=temp->next;}
67 }
68 void concatenation(Node *Flist,Node *Slist){
69     bool flag=false;
70     if( Flist != NULL && Slist!= NULL ){
71         if (Flist->next == NULL)
72             Flist->next = Slist;
73         else
74             concatenation(Flist->next,Slist);
75         flag=true;}
76     else{
77         if(flag==false){
78             cout<<"The given two input lists are empty.";
79         }}}
80 bool emptiness(){
81     bool flag=false;
82     if(head==NULL){
83         flag=true;
84         return flag;
85     }
86 }

```

```

87 void concatenate(Node* object1head, Node* object2head){
88     Node* temp;
89     temp = object1head;
90     while(temp!=NULL){
91         temp=temp->next;
92     }
93     temp->next=object2head;
94 }
95 void merging_order(Node* object1, Node* object2){
96     Node* temp;
97     temp=object1;
98     while(temp!=NULL){
99         temp=temp->next;
100     }
101     temp->next=object2;
102     temp=head;
103     while(temp!=NULL){
104         if(temp->data>temp->next->data){
105             temp->data=temp->next->data;}}
106 }

```

```
107 void summing(){
108     Node* temp;
109     temp=head;
110     int average=0;
111     int summed=0;
112     int counter=0;
113     while(temp!=NULL){
114         summed+=temp->data;
115         counter++;
116         temp=temp->next;
117     }
118     average=summed/counter;
119 }
120 void copied(Node* object1){
121     Node* temp;
122     temp=object1;
123     int length=0;
124     int i=0;
125     while(temp!=NULL){
126         length++;
127         temp=temp->next;
128     }
129     while(length!=0){
130         if(i!=length){
131             Node* newbie;
132             newbie->data=temp->data;
133             length--;
134             i++;
135         }
136     }
```

```
137 void insertmiddle(int val){
138     Node* temp;
139     temp=head;
140     int i=0;
141     int counter=0;
142     while(temp!=head){
143         counter++;
144         temp=temp->next;
145     }
146     temp=head;
147     while(i!=counter/2){
148         temp=temp->next;
149     }
150     temp->data=val;
151 }
152 void movefirst(){
153     Node* first=new Node(0);
154     Node* second= new Node(0);
155     Node* temp;
156     temp=head;
157     while(temp!=NULL){
158         temp=temp->next;
159     }
160     second->data=temp->data;
161     first->data=head->data;
162     if(head!=NULL){
163         head->data=second->data;
164         temp->data=first->data;
165     }
```

```

166     void sorted(Node* object1){
167         Node* temp;
168         temp=object1;
169         while(temp!=NULL){
170             if(temp->data>temp->next->data)
171                 temp->data=temp->next->data;
172             temp=temp->next;
173         }
174     }

```

```

186 void RemoveDuple(Node*object){
187     //object.sorted();
188     Node* temp, *worker;
189     temp=head;
190     while(temp!=NULL){
191         temp=temp->next;
192         if(temp->next->data!=temp->data){
193             worker=temp;
194             temp=temp->next;}
195         else{
196             worker=temp;
197             temp=temp->next;
198             delete worker;}
199     }
200 }
201 };

```

```

201 bool isPalin(Node* head){
202     Node* slow= head;
203     while(slow != NULL){
204         slow = slow->next;
205     }
206     while(head != NULL ){
207         int i;
208         if(head -> data != i){
209             return false;}
210         head=head->next;}
211     return true;}

```



```
213 - int main(){
214     LinkList *obj1;
215     LinkList *obj2;
216     for (int x=0; x<15;x++)
217 - {
218         int randomNumber = (rand() % 100);
219         //obj1.insertionatend(randomNumber);
220         //obj1.insertionatend(randomNumber);
221         //obj1.insertionatend(randomNumber);
222         //obj1.insertionatend(randomNumber);
223         //obj1.insertionatend(randomNumber);
224         obj1.insertionatmid(44);
225         obj1.insertionatmid(1);
226         obj1.insertionatmid(2);
227         obj1.insertionatmid(8);
228         obj1.insertionatmid(9);
229         obj1.insertionatmid(6);
230
231         obj2.insertionatmid(13);
232         obj2.insertionatmid(8);
233         obj2.insertionatmid(15);
234         obj2.insertionatmid(23);
235         obj2.insertionatmid(25);
236         obj2.insertionatmid(6);
237
238         Node*ptr1,*pt2;
239         ptr1=&obj1;
240         ptr2=&obj2;
241         concatenation(ptr1, ptr2);
242
243         obj1.printed();
244         obj2.printed();
```

```

231     obj2.insertionatmid(13);
232     obj2.insertionatmid(8);
233     obj2.insertionatmid(15);
234     obj2.insertionatmid(23);
235     obj2.insertionatmid(25);
236     obj2.insertionatmid(6);
237
238     Node*ptr1,*pt2;
239     ptr1=&obj1;
240     ptr2=&obj2;
241     concatenation(ptr1, ptr2);
242
243     obj1.printed();
244     obj2.printed();
245
246     obj1.summing();
247     obj2.summing();
248
249     obj1.sorted();
250
251
252
253
254 }

```

The outputs are:

44	1	2	9	8	7	15	23	NULL
----	---	---	---	---	---	----	----	------

Output

13	8	15	23	25	6
----	---	----	----	----	---

Output

The Lists after concatenation are: 44,1,2,8,9,6,13,8,15,23,25,6
--

Output

The values are: 13,8,15,23,25,6

Output

The values are;
13,8,15,23,25,6

Output

The Summed Values are:
70

Output

The Sum of values are:
90

Output

Sorted Values are:
1,2,6,8,9,44