DIGITAL ASSIGNMENT 2

NAME-AKSHAT SRIVASTAV
REG. NO-19BCE0811

Write down a C/C++/C#/JAVA/Python/MATLAB/R Program to Implement Doubly Circular Linked List.

Your program must have following functions:

- **InsertNode: insert a new node at given position**
- **DeleteNode: delete a node at given position**
- **FindNode: find a node with a given value**
- **DisplayList: print all the nodes in the list**

General Instructions:

1) You have to submit your program in VTOP (submit it as a PDF file - don't upload
2) source files).
3) Last Date of Submission: 28th February 2020.
4) Please do include some COMMENTS into your code to make it easier to understand.
5) Your program is correct does not necessarily mean that you will get full marks.
6) Program written C/C++ language is preferable.
7) Please do not plagiarize someone else's work.

```
//CIRCULAR DOUBLE LINKED LIST
#include<iostream>
#include<cstdio>
#include<cstdlib>
using namespace std;
struct nod {
   int info;
   struct nod *n;//creating a node
   struct nod *p;//as double linked list creating another node
}*start, *last;
int count = 0;
class circulardoublylist {//creating a class
   public:
      nod *create_node(int);
      void insert_begin();//declaring all the functions
      void insert_end();
      void insert_pos();
      void delete_pos();
      void search();
      void update();
      void display();
      void reverse();
      circulardoublylist() {
         start = NULL;//initializing the starting value to null
      last = NULL;//initializing the last value to null 26
      }
};
int main() {
   int c;
   circulardoublylist cdl;
   while (1){ //perform switch operation {
      cout<<"1.Insert at Beginning"<<endl;//providing the options of what to do
      cout<<"2.Insert at End"<<endl;
      cout<<"3.Insert at Position"<<endl;
      cout<<"4.Delete at Position"<<endl;
      cout<<"5.Update Node"<<endl;
      cout<<"6.Search Element"<<endl;
      cout<<"7.Display List"<<endl;
      cout<<"8.Reverse List"<<endl;
      cout<<"9.Exit"<<endl;
      cout<<"Enter your choice : ";
      cin>>c;
      switch(c) {//assigning the functions
         case 1:
            cdl.insert_begin();
         break;
         case 2:
            cdl.insert_end();
         break;
         case 3:
            cdl.insert_pos();
         break;
         case 4:
            cdl.delete_pos();
         break;
         case 5:
            cdl.update();
         break;
         case 6:
            cdl.search();
         break;
         case 7:
            cdl.display();
         break;
         case 8:
            cdl.reverse();
```

```
67            break;
68            case 9:
69               exit(1);
70            default:
71         cout<<"Wrong choice"<<endl; 72
           }
73      }
74      return 0;
75   }
76   nod* circulardoublylist::create_node(int v) {
77      count++;
78      struct nod *t;
79      t = new(struct nod);
80      t->info = v;
81      t->n = NULL;
82      t->p = NULL;
83      return t;
84   }
85   void circulardoublylist::insert_begin() {//calling the class to extract the elements
86      int v;
87      cout<<endl<<"Enter the element to be inserted: ";
88      cin>>v;
89      struct nod *t;
90      t = create_node(v);
91      if (start == last && start == NULL) {//creating a condition and pointing the pointer
92         cout<<"Element inserted in empty list"<<endl;
93         start = last = t;
94         start->n = last->n = NULL;
95         start->p = last->p = NULL;
96      } else {
97         t->n = start;
98         start->p = t;
99         start = t;
100        start->p = last;
101        last->n = start;
102        cout<<"Element inserted"<<endl;
103     }
104  }
105  void circulardoublylist::insert_end() {//insertion at the end
106     int v;
107     cout<<endl<<"Enter the element to be inserted: ";
108     cin>>v;
109     struct nod *t;
110     t = create_node(v);
111     if (start == last && start == NULL) {
112        cout<<"Element inserted in empty list"<<endl;
113        start = last = t;
114        start->n= last->n = NULL;
115        start->p = last->p= NULL;
116     } else {
117        last->n= t;
118        t->p= last;
119        last = t;
120        start->p = last;
121        last->n= start;
122     }
123  }
124  void circulardoublylist::insert_pos() {//insertion at some another position
125     int v, pos, i;
126     cout<<endl<<"Enter the element to be inserted: ";
127     cin>>v;
128     cout<<endl<<"Enter the position of element inserted: ";
129     cin>>pos;
130     struct nod *t, *s, *ptr;
131     t = create_node(v);
132     if (start == last && start == NULL) {
```

```
133         if (pos == 1) {
134            start = last = t;
135            start->n = last->n = NULL;
136            start->p = last->p = NULL;
137         } else {
138            cout<<"Position out of range"<<endl;
139            count--;
140            return;
141         }
142      } else {
143         if (count < pos) {
144            cout<<"Position out of range"<<endl;
145            count--;
146            return;
147         }
148         s = start;
149         for (i = 1;i <= count;i++) {
150            ptr = s;
151            s = s->n;
152            if (i == pos - 1) {
153               ptr->n = t;
154               t->p= ptr;
155               t->n= s;
156               s->p = t;
157               cout<<"Element inserted"<<endl;
158               break;
159            }
160         }
161      }
162  }
163   void circulardoublylist::delete_pos() {//deletion of any element
164      int pos, i;
165      nod *ptr, *s;
166      if (start == last && start == NULL) {
167         cout<<"List is empty, nothing to delete"<<endl;
168         return;
169      }
170      cout<<endl<<"Enter the position of element to be deleted: ";
171      cin>>pos;
172      if (count < pos) {
173         cout<<"Position out of range"<<endl;
174         return;
175      }
176      s = start;
177      if(pos == 1) {
178         count--;
179         last->n = s->n;
180         s->n->p = last;
181         start = s->n;
182         free(s);
183         cout<<"Element Deleted"<<endl;
184         return;
185      }
186      for (i = 0;i < pos - 1;i++ ) {
187         s = s->n;
188         ptr = s->p;
189      }
190      ptr->n = s->n;
191      s->n->p = ptr;
192      if (pos == count) {
193         last = ptr;
194      }
195      count--;
196      free(s);
197      cout<<"Element Deleted"<<endl;
198  }
```

```
199 void circulardoublylist::update() {//updating the double linked list
200    int v, i, pos;
201    if (start == last && start == NULL) {
202       cout<<"The List is empty, nothing to update"<<endl;
203       return;
204    }
205    cout<<endl<<"Enter the position of node to be updated: ";
206    cin>>pos;
207    cout<<"Enter the new value: ";
208    cin>>v;
209    struct nod *s;
210    if (count < pos) {
211       cout<<"Position out of range"<<endl;
212       return;
213    }
214    s = start;
215    if (pos == 1) {
216       s->info = v;
217       cout<<"Node Updated"<<endl;
218       return;
219    }
220    for (i=0;i < pos - 1;i++) {
221       s = s->n;
222    }
223    s->info = v;
224    cout<<"Node Updated"<<endl;
225 }
226 void circulardoublylist::search() {//searching a node/element
227    int pos = 0, v, i;
228    bool flag = false;
229    struct nod *s;
230    if (start == last && start == NULL) {
231       cout<<"The List is empty, nothing to search"<<endl;
232       return;
233    }
234    cout<<endl<<"Enter the value to be searched: ";
235    cin>>v;
236    s = start;
237    for (i = 0;i < count;i++) {
238       pos++;
239       if (s->info == v) {
240          cout<<"Element "<<v<<" found at position: "<<pos<<endl;
241          flag = true;
242       }
243       s = s->n;
244    }
245    if (!flag)
246       cout<<"Element not found in the list"<<endl;
247 }
248 void circulardoublylist::display() {//displaying the double circular linked list
249    int i;
250    struct nod *s;
251    if (start == last && start == NULL) {
252       cout<<"The List is empty, nothing to display"<<endl;
253       return;
254    }
255    s = start;
256    for (i = 0;i < count-1;i++) {
257       cout<<s->info<<"<->";
258       s = s->n;
259    }
260    cout<<s->info<<endl;
261 }
262 void circulardoublylist::reverse() {//reversing the whole list
263    if (start == last && start == NULL) {
264       cout<<"The List is empty, nothing to reverse"<<endl;
```

```
265         return;
266      }
267      struct nod *p1, *p2;
268      p1 = start;
269      p2 = p1->n;
270      p1->n = NULL;
271      p1->p= p2;
272      while (p2 != start) {
273         p2->p = p2->n;
274         p2->n = p1;
275         p1 = p2;
276         p2 = p2->p;
277      }
278      last = start;
279      start = p1;
280      cout<<"List Reversed"<<endl;
281      }
```