PRN: 2020BTEIT00041

Name: Om Vivek Gharge

Q.) Circular Linked List

CODE:

```cpp
#include <bits/stdc++.h>
using namespace std;

class Node{
public:
    int data;
    Node *next;

    Node(int d){
        data = d;
        next = NULL;
    }
};

class Head{
    public:
    int count;
    Node *first;
    Node *last;

    Head(int c, Node *f, Node* l){
        this->count = c;
        this->first = f;
        this->last = l;
    }
};

// Function to insert at head of circular linked list
void insertAtHead(Head *head, int data){
    Node *newNode = new Node(data);

    if(head->count == 0){
        head->first = newNode;
        head->last = newNode;

        head->first->next = head->last;
        head->last->next = head->first;
    }
    else{
```

```
        newNode->next = head->first;
        head->first = newNode;
        head->last->next = head->first;
    }
    head->count++;
}

// Function to insert at tail of circular linked list
void insertAtTail(Head *head, int data){
    Node *newNode = new Node(data);

    if(head->count == 0){
        head->first = newNode;
        head->last = newNode;

        head->first->next = head->last;
        head->last->next = head->first;
    }
    else{
        head->last->next = newNode;
        head->last = newNode;
        head->last->next = head->first;
    }
    head->count++;
}

// Fucntion to insert at any position of circular linked list
void insertAfter(Head *head, int data, int pos){
    Node *newNode = new Node(data);
    Node *temp = head->first;
    Node *prev = head->first;

    if(pos == 1){
        insertAtHead(head, data);
    }
    else if(pos == head->count + 1){
        insertAtTail(head, data);
```

```
    }
    else{
        for(int i = 1; i < pos; i++){
            prev = temp;
            temp = temp->next;
        }
        prev->next = newNode;
        newNode->next = temp;
        head->count++;
    }
}

// Function to delete a node by value from circular linked list
void deleteByValue(Head *head, int data){
    Node *temp = head->first;
    Node *prev = head->first;

    if(head->count == 0){
        cout << "List is empty" << endl;
    }
    else if(head->count == 1){
        if(head->first->data == data){
            head->first = NULL;
            head->last = NULL;
            head->count--;
        }
        else{
            cout << "Element not found" << endl;
        }
    }
    else{
        if(head->first->data == data){
            head->first = head->first->next;
            head->last->next = head->first;
            head->count--;
        }
        else{
            while(temp->data != data){
```

```
                prev = temp;
                temp = temp->next;
            }
            if(temp->data == data){
                prev->next = temp->next;
                head->count--;
            }
            else{
                cout << "Element not found" << endl;
            }
        }
    }
}

// Function to delete a node by position from circular linked list
void deleteByPos(Head *head, int pos){
    Node *temp = head->first;
    Node *prev = head->first;

    if(head->count == 0){
        cout << "List is empty" << endl;
    }
    else if(head->count == 1){
        if(pos == 1){
            head->first = NULL;
            head->last = NULL;
            head->count--;
        }
        else{
            cout << "Element not found" << endl;
        }
    }
    else{
        if(pos == 1){
            head->first = head->first->next;
            head->last->next = head->first;
            head->count--;
```

```
        }
        else if(pos == head->count){
            while(temp->next != head->first){
                prev = temp;
                temp = temp->next;
            }
            prev->next = head->first;
            head->last = prev;
            head->count--;
        }
        else{
            for(int i = 1; i < pos; i++){
                prev = temp;
                temp = temp->next;
            }
            prev->next = temp->next;
            head->count--;
        }
    }
}

// Function to search a node by value from circular linked list
void searchByValue(Head *head, int data){
    Node *temp = head->first;

    if(head->count == 0){
        cout << "List is empty" << endl;
    }
    else{
        for(int i=0; i<head->count; i++){
            if(temp->data == data){
                cout << "Element found at position " << i+1 << endl;
                return;
            }
            else{
                temp = temp->next;
            }
        }
    }
}

// Function to print circular linked list
void printList(Head *head){
    Node *temp = head->first;

    if(head->count == 0){
        cout << "List is empty" << endl;
    }
    else{
        for(int i=0; i<head->count; i++){
            cout << temp->data << " ";
            temp = temp->next;
        }
        cout << endl;
    }
}

int main(){
    Head* h = new Head(0, NULL, NULL);

    int opt, data, index;
    char choice;
    while(1){
        cout<<"\nMENU\n a. Add directly to LinkedList.\n b. Use functions.\n";
        cout<<"Choose: ";
        cin>>choice;

        switch (choice){

        case 'a':
            int num;
            cout<<"Enter the data to add in LinkedList (enter '-1' if you want stop): ";
            while(1){
                cin>>num;
                if(num == -1) break;
```

```
227
228                 Node* addNode = new Node(num);
229                 if(h->count == 0){
230                     h->first = addNode;
231                     h->last = addNode;
232                     addNode->next = h->first;
233
234                     h->count++;
235                 }
236                 else{
237                     h->last->next = addNode;
238                     h->last = addNode;
239                     addNode->next = h->first;
240
241                     h->count++;
242                 }
243             }
244             break;
245
246             case 'b':
247             cout<<"\nMENU 1.0\n 1.Add at head\n 2.Add at tail\n 3.Add after\n 4.Delete (by Value)\n 5.Delete (by Index))\n 6.Search\n 7.Display the List\n 8.Exit\n";
248             cout<<"Enter your option: ";
249
250             cin>>opt;
251             cout<<"\n";
252             if(opt>7) break;
253
254             switch(opt){
255
256                 case 1:
257                     cout<<"Enter data to add: ";
258                     cin>>data;
259                     cout<<"Adding data...\n";
260                     insertAtHead(h, data);
261                     cout<<"\n";
262                     break;
263
264                 case 2:
265                     cout<<"Enter data to add: ";
266                     cin>>data;
267                     cout<<"Adding data...\n";
268                     insertAtTail(h, data);
269                     cout<<"\n";
270                     break;
271
272                 case 3:
273                     cout<<"Enter the index: ";
274                     cin>>index;
275                     cout<<"Enter data to add: ";
276                     cin>>data;
277                     cout<<"Adding data...\n";
278                     insertAfter(h, data, index);
279                     cout<<"\n";
280                     break;
281
282                 case 4:
283                     cout<<"Enter data to delete: ";
284                     cin>>data;
285                     cout<<"Deleting data...\n";
286                     deleteByValue(h, data);
287                     cout<<"\n";
288                     break;
289
290                 case 5:
291                     cout<<"Enter the index: ";
292                     cin>>index;
293                     cout<<"Deleting data...\n";
294                     deleteByPos(h, index);
295                     cout<<"\n";
296                     break;
297
298                 case 6:
299                     cout<<"Enter data to search: ";
300                     cin>>data;
301                     cout<<"Searching data...\n";
302                     searchByValue(h, data);
303                     cout<<endl;
```

```
304                     break;
305
306                 case 7:
307                     cout<<"Displaying the LinkedList: ";
308                     printList(h);
309                     cout<<"\n";
310                     break;
311
312                 default:
313                     break;
314
315             }
316
317         default:
318             break;
319         }
320     }
321
322     return 0;
323 }
```

OUTPUT:

```
MENU
 a. Add directly to LinkedList.
 b. Use functions.
Choose: a
Enter the data to add in LinkedList (enter '-1' if you want stop): 1 2 3 4 5
-1

MENU
 a. Add directly to LinkedList.
 b. Use functions.
Choose: b

MENU 1.0
 1.Add at head
 2.Add at tail
 3.Add after
 4.Delete (by Value)
 5.Delete (by Index))
 6.Search
 7.Display the List
 8.Exit
Enter your option: 1

Enter data to add: 0
Adding data...


MENU
 a. Add directly to LinkedList.
 b. Use functions.
Choose: b

MENU 1.0
 1.Add at head
 2.Add at tail
 3.Add after
 4.Delete (by Value)
 5.Delete (by Index))
 6.Search
 7.Display the List
 8.Exit
Enter your option: 7
```

```
Displaying the LinkedList: 0 1 2 3 4 5


MENU
 a. Add directly to LinkedList.
 b. Use functions.
Choose: b

MENU 1.0
 1.Add at head
 2.Add at tail
 3.Add after
 4.Delete (by Value)
 5.Delete (by Index))
 6.Search
 7.Display the List
 8.Exit
Enter your option: 4

Enter data to delete: 5
Deleting data...


MENU
 a. Add directly to LinkedList.
 b. Use functions.
Choose: b

MENU 1.0
 1.Add at head
 2.Add at tail
 3.Add after
 4.Delete (by Value)
 5.Delete (by Index))
 6.Search
 7.Display the List
 8.Exit
Enter your option: 7

Displaying the LinkedList: 0 1 2 3 4
```

ALGORITHM:

Circular Linked List

ALGORITHMS:

1.Insertion of node in the circular linked list at the beginning

Step 1:

IF PTR = NULL

Write OVERFLOW

Go to Step 11

[END OF IF]

Step 2: SET NEW_NODE = PTR

Step 3: SET PTR = PTR -> NEXT

Step 4: SET NEW_NODE -> DATA = VAL

Step 5: SET TEMP = HEAD

Step 6: Repeat Step 8 while TEMP -> NEXT != HEAD

Step 7: SET TEMP = TEMP -> NEXT

[END OF LOOP]

Step 8: SET NEW_NODE -> NEXT = HEAD

Step 9: SET TEMP → NEXT = NEW_NODE

Step 10: SET HEAD = NEW_NODE

Step 11: EXIT

2.Insertion of node in the circular linked list at the end

Step 1: IF PTR = NULL

Write OVERFLOW

Go to Step 1

[END OF IF]

Step 2: SET NEW_NODE = PTR

Step 3: SET PTR = PTR -> NEXT

Step 4: SET NEW_NODE -> DATA = VAL

Step 5: SET NEW_NODE -> NEXT = HEAD

Step 6: SET TEMP = HEAD

Step 7: Repeat Step 8 while TEMP -> NEXT != HEAD

Step 8: SET TEMP = TEMP -> NEXT

[END OF LOOP]

Step 9: SET TEMP -> NEXT = NEW_NODE

Step 10: EXIT

3.Deletion of node in the circular linked list at the beginning

Step 1: IF HEAD = NULL

Write UNDERFLOW

Go to Step 8

[END OF IF]

Step 2: SET PTR = HEAD

Step 3: Repeat Step 4 while PTR → NEXT != HEAD

Step 4: SET PTR = PTR → next

[END OF LOOP]

Step 5: SET PTR → NEXT = HEAD → NEXT

Step 6: FREE HEAD

Step 7: SET HEAD = PTR → NEXT

Step 8: EXIT

4.Deletion of node in the circular linked list at the end

Step 1: IF HEAD = NULL

Write UNDERFLOW

Go to Step 8

[END OF IF]

Step 2: SET PTR = HEAD

Step 3: Repeat Steps 4 and 5 while PTR -> NEXT != HEAD

Step 4: SET PREPTR = PTR

Step 5: SET PTR = PTR -> NEXT

[END OF LOOP]

Step 6: SET PREPTR -> NEXT = HEAD

Step 7: FREE PTR

Step 8: EXIT

5.Searching in circular linked list

Step 1: SET PTR = HEAD

Step 2: Set I = 0

STEP 3: IF PTR = NULL

WRITE "EMPTY LIST"

GOTO STEP 8

END OF IF

STEP 4: IF HEAD → DATA = ITEM

WRITE i+1 RETURN [END OF IF]

STEP 5: REPEAT STEP 5 TO 7 UNTIL PTR->next != head

STEP 6: if ptr → data = item

write i+1

RETURN

End of IF

STEP 7: I = I + 1

STEP 8: PTR = PTR → NEXT

[END OF LOOP]

STEP 9: EXIT

6.Traversing in circular linked list

STEP 1: SET PTR = HEAD

STEP 2: IF PTR = NULL

WRITE "EMPTY LIST"

GOTO STEP 8

END OF IF

STEP 4: REPEAT STEP 5 AND 6 UNTIL PTR → NEXT != HEAD

STEP 5: PRINT PTR → DATA

STEP 6: PTR = PTR → NEXT

[END OF LOOP]

STEP 7: PRINT PTR→ DATA

STEP 8: EXIT