PRN: 2020BTEIT00041

Name: Om Vivek Gharge

Q1. Singly LinkedList

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

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

    Node(int data){
        this->data = data;
        this->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;
    }
};

void InsertAtFront(Head *h, int data){

    Node* new_node = new Node(data);

    if(h->count==0){
        h->first = new_node;
        h->last = new_node;
        h->count++;

        return;
    }
    else{
        new_node->next = h->first;
        h->first = new_node;
        h->count++;

        return;
    }
}

void InsertAtBack(Head *h, int data){
```

```
49
50      Node* new_node = new Node(data);
51      if(h->count==0){
52          h->first = new_node;
53          h->last = new_node;
54          h->count++;
55
56          return;
57      }
58      else{
59          Node* p = h->first;
60
61          for(int i=0; i<h->count; i++){
62
63              if(p->next==NULL){
64                  p->next = new_node;
65                  h->count++;
66                  return;
67              }
68              p = p->next;
69          }
70      }
71  }
72
73  void insertAfter(Head *h, int data, int location){
74
75      if(location == 0){
76          InsertAtFront(h, data);
77          return;
78      }
79      else{
80          Node* new_node = new Node(data);
81
82          Node* p = h->first;
83
84          for(int i=1; i<h->count; i++){
85
86              if(i==location){
87                  new_node->next = p->next;
88                  p->next = new_node;
89                  h->count++;
90                  return;
91              }
92              p = p->next;
93          }
94      }
95
96  }
```

```cpp

void delNode_byVal(Head* h, int val){

    if(h->count==0){
        cout<<"List is empty, can't delete."<<"\n";
        return;
    }

    Node* p = h->first;

    //  To Del head
    if(p->data==val){
        h->first = p->next;
        p->next = NULL;
        free(p);
        h->count--;

        return;
    }

    Node* q;
    for(int i=0; i<h->count; i++){

        if(p->data==val){
            q->next = p->next;
            p->next = NULL;
            free(p);
            h->count--;

            return;
        }

        q = p;
        p = p->next;
    }
}

void delNodeAt(Head* h, int location){

    if(h->count==0){
        cout<<"List is empty, can't delete."<<"\n";
        return;
    }

    Node* p = h->first;

    //  To Del head
    if(location==0){
```

```
145             h->first = p->next;
146             p->next = NULL;
147             free(p);
148             h->count--;
149 
150             return;
151         }
152 
153         Node* q;
154         for(int i=1; i<h->count; i++){
155 
156             if(i==location){
157                 q->next = p->next;
158                 p->next = NULL;
159                 free(p);
160                 h->count--;
161 
162                 return;
163             }
164 
165             q = p;
166             p = p->next;
167         }
168     }
169 
170     bool searchList(Head* h, int key){
171 
172         Node* p = h->first;
173 
174         for(int i=0; i<h->count; i++){
175 
176             if(p->data == key){
177                 return true;
178             }
179 
180             p = p->next;
181         }
182 
183         return false;
184     }
185 
186     void printList(Head *h){
187         Node* p = new Node(0);
188 
189         p = h->first;
190 
191         for(int i=0; i<h->count; i++){
192             cout<<p->data<<" ";
```

```
        p = p->next;
    }
}

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<<"Choice: ";
        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;

                Node* addNode = new Node(num);
                if(h->count == 0){
                    h->first = addNode;
                    h->last = addNode;
                    addNode->next = h->first;

                    h->count++;
                }
                else{
                    h->last->next = addNode;
                    h->last = addNode;
                    addNode->next = h->first;

                    h->count++;
                }
            }
            break;

            case 'b':
            cout<<"\nMENU 1.Use 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";
            cout<<"Enter your option: ";

            cin>>opt;
            cout<<"\n";
              if(opt>7) break;

              switch(opt){

                  case 1:
                      cout<<"Enter data to add: ";
                      cin>>data;
                      cout<<"Adding data...\n";
                      InsertAtFront(h, data);
                      cout<<"\n";
                      break;

                  case 2:
                      cout<<"Enter data to add: ";
                      cin>>data;
                      cout<<"Adding data...\n";
                      InsertAtBack(h, data);
                      cout<<"\n";
                      break;

                  case 3:
                      cout<<"Enter the index: ";
                      cin>>index;
                      cout<<"Enter data to add: ";
                      cin>>data;
                      cout<<"Adding data...\n";
                      InsertAfter(h, data, index);
                      cout<<"\n";
                      break;

                  case 4:
                      cout<<"Enter data to delete: ";
                      cin>>data;
                      cout<<"Deleting data...\n";
                      delNode_byVal(h, data);
                      cout<<"\n";
                      break;

                  case 5:
                      cout<<"Enter the index: ";
                      cin>>index;
                      cout<<"Deleting data...\n";
                      delNodeAt(h, index);
                      cout<<"\n";
                      break;
```

```
285
286                 case 6:
287                     cout<<"Enter data to search: ";
288                     cin>>data;
289                     cout<<"Searching data...\n";
290                     searchList(h, data);
291                     cout<<endl;
292                     break;
293
294                 case 7:
295                     cout<<"Displaying the LinkedList: ";
296                     printList(h);
297                     cout<<"\n";
298                     break;
299
300                 default:
301                     break;
302
303             }
304
305         default:
306             break;
307         }
308     }
309
310     return 0;
311 }
```

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
```

ALGORITHM:

ALGORITHMS:

1.Insertion In Singly Linked List A t Beginning:

Step 1: IF PTR = NULL

Write OVERFLOW

Go to Step 7

[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 HEAD = NEW_NODE

Step 7: EXIT

2.Insertion in singly linked list after specified Node

STEP 1: IF PTR = NULL

WRITE OVERFLOW

GOTO STEP 12

END OF IF

STEP 2: SET NEW_NODE = PTR

STEP 3: NEW_NODE → DATA = VAL

STEP 4: SET TEMP = HEAD

STEP 5: SET I = 0

STEP 6: REPEAT STEP 5 AND 6 UNTIL I<loc< li=""></loc<>

STEP 7: TEMP = TEMP → NEXT

STEP 8: IF TEMP = NULL

WRITE "DESIRED NODE NOT PRESENT"

GOTO STEP 12

END OF IF

END OF LOOP

STEP 9: PTR → NEXT = TEMP → NEXT

STEP 10: TEMP → NEXT = PTR

STEP 11: SET PTR = NEW_NODE

STEP 12: EXIT

3. Insertion in singly linked list at the tail:

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 = NULL

Step 6: SET PTR = HEAD

Step 7: Repeat Step 8 while PTR - > NEXT != NULL

Step 8: SET PTR = PTR - > NEXT

[END OF LOOP]

Step 9: SET PTR - > NEXT = NEW_NODE

Step 10: EXIT

4. Deletion in singly linked list at beginning:

Step 1: IF HEAD = NULL

Write UNDERFLOW

Go to Step 5

[END OF IF]

Step 2: SET PTR = HEAD

Step 3: SET HEAD = HEAD -> NEXT

Step 4: FREE PTR

Step 5: EXIT

5. Deletion in singly linked list after the specified node :

STEP 1: IF HEAD = NULL

WRITE UNDERFLOW

GOTO STEP 10

END OF IF

STEP 2: SET TEMP = HEAD

STEP 3: SET I = 0

STEP 4: REPEAT STEP 5 TO 8 UNTIL I<loc< li=""></loc<>

STEP 5: TEMP1 = TEMP

STEP 6: TEMP = TEMP → NEXT

STEP 7: IF TEMP = NULL

WRITE "DESIRED NODE NOT PRESENT"

GOTO STEP 12

END OF IF

STEP 8: I = I+1

END OF LOOP

STEP 9: TEMP1 → NEXT = TEMP → NEXT

STEP 10: FREE TEMP

STEP 11: EXIT

6. Deletion in singly 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!= NULL

Step 4: SET PREPTR = PTR

Step 5: SET PTR = PTR -> NEXT

[END OF LOOP]

Step 6: SET PREPTR -> NEXT = NULL

Step 7: FREE PTR

Step 8: EXIT

7. Searching in singly 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: REPEAT STEP 5 TO 7 UNTIL PTR != NULL

STEP 5: if PTR → DATA = ITEM

write i+1

End of IF

STEP 6: I = I + 1

STEP 7: PTR = PTR → NEXT

[END OF LOOP]

STEP 8: EXIT

8. Reversing Singly Linked List:

STEP 1:Take 3 nodes as Node ptrOne,Node ptrTwo, Node prevNode

STEP 2:Initialize them as ptrOne = head; ptrTwo=head.next, prevNode = null.

STEP 3:Call reverseRecursion(head,head.next,null)

STEP 4: Reverse the ptrOne and ptrTwo

STEP 5: Make a recursive call for reverseRecursion(ptrOne.next,ptrTwo.next,null)