Shreshtha Aggarwal
1BM19CS155

Date
Page 58

14/12/20

Lab - 9 (Week 10)

Q-> WAP of doubly linked list : a) creation b) insertion c) delete front d) delete rear e) Search f) insertion before & after key g) Delete repeating occurences.

```
# include <stdio.h>
# include <stdlib.h>
struct node { int info;
    struct node *rlink;
    struct node *llink;
}; typedef struct node *Node;
Node getnode () { Node x;
    x = (Node) malloc (sizeof(struct node));
    if (x == NULL) { printf("Memory full \n");
        exit (0); }
    return x; }
void freenode (Node x) { free (x); }
Node dinsert_rear (Node head, int item) {
    Node temp, cur;
    temp = getnode ();
    temp -> rlink = NULL;
    temp -> llink = NULL;
    temp -> info = item;
    cur -> rlink = temp;
    head -> llink = temp;
    temp -> rlink = head;
```

```
        head -> info = head -> info +1;
        return head; }
Node  dinsert_front (int item, Node head) {
        Node temp, cur;
        temp = getnode ();
        temp -> info = item;
        cur = head -> rlink;
        head -> rlink = temp;
        temp -> llink = head;
        temp -> rlink = cur;
        cur -> llink = temp;
        return head;
}
Node insert_leftpos (int item, Node head) {
        Node temp, cur, prev;
        if (head -> rlink == head) {
              printf (" List empty \n");
              return head;
        }
        cur = head -> rlink;
        while (cur != head) { if (item == cur->info)
                                   break;
              cur = cur -> rlink; }
        if (cur == head) { printf (" Not found\n");
              return head; }
        prev = cur -> llink;
        printf(" Enter towards left of %d = ",
                item);
        temp = getnode ();
        ~~scanf = ( getnode~~
        scanf (" %d ", & temp -> info);
        prev -> rlink = temp;
        temp -> llink = prev;
        cur -> llink = temp;
        temp -> rlink = cur;
        return head;
}
NODE insert_rightpos (int item, NODE head){
        NODE temp, cur, prev;
        if (head -> rlink == head) {
                printf (" List empty \n" );
                return head; }
        cur = head -> rlink;
        while ( cur != head) {
                if (item == cur -> info) {
                        break; }
                cur = cur -> rlink; }
        if ( cur == head) {
                printf (" Key Not found \n");
                return head; }
        prev = cur -> rlink;
        printf (" Enter item :- ");
        temp = getnode ();
        scanf ("%d", & temp -> info);
        prev -> llink = temp;
        temp -> rlink = prev;
        cur -> rlink = temp;
        temp -> llink = cur;
        return head; }.
NODE ddelete_front (NODE head) {
        NODE cur, next;
        if (head -> rlink == head) {
            printf ("List Empty \n");
            return head;
        }
        cur = head -> rlink;
        next = cur -> rlink;
        head -> rlink = next;
        next -> llink = head;
        printf (" Deleted Key = %d", cur->info);
        freenode (cur);
        return head;.
}
NODE ddelete_rear (NODE head){
        NODE cur, prev;
```

Date
Page 62

```
            if (head -> rlink == head) {
                    printf (" List empty \n");
                    return head; }
            cur = head -> llink;
            prev = cur -> llink;
            head -> llink = prev;
            prev -> rlink = head;
            printf (" Del = %d ", cur -> info ");
            freenode (cur);
            return head;
}
void display (NODE head) {
            NODE temp;
            if (head -> rlink == head) {
                    printf (" List empty .. \n");
                    return; }
            for (temp = head -> rlink ; temp !=
                 head ; temp = temp -> rlink) {
                 printf ("%d \n", temp -> info); },
}
void search (int item, NODE head) {
        NODE temp;
        int j = 0;
        temp = head -> rlink;
        while ( temp != head) {  j++;
```

```
    if (temp -> info == item) {
            printf ("Item at %d", j);
            return;
    temp = temp -> rlink; }
  printf ("Search Unsuccessful"); }
NODE delete_all (int item, NODE head) {
    NODE prev, cur, next;
    int count = 0; j = 0;
    if (head -> rlink == head) {
            printf ("Not found");
            return head; }
    cur = head -> rlink;
    while (cur != head) {
         if (item != cur -> info) {
              cur = cur -> rlink; }
         else { j++;
              if (j == 1) { cur = cur -> rlink; }
              if (item == cur -> info) {
                   count++;
                   prev = cur -> llink;
                   next = cur -> rlink;
                   prev -> rlink = next;
                   next -> llink = prev;
                   freenode (cur);
                   cur = next; } } }
```

```
        if (count == 0) {
                printf ("No duplicates \n"); }
        else
                printf ("Ref key at %d & del,"cont);
}
int main () {
        int item, choice;
        NODE head;
        head = getnode ();
        head -> rlink = head;
        head -> llink = head;
        for (;;) {
                printf("\n1. Insert rear \n2. Insert front
                \n3. Insert left \n4. Insert right \n5.
                Delete front \n6. Delete rear \n7.
                Delete dup \n8. Search \n9. Display");
            printf (" Enter choice : ");
            scanf ("%d", & choice);
            switch (choice) {
                case 1: printf (" Enter item : ");
                        scanf ("%d", & item);
                        head = dinsert_rear (head, item);
                        break;
                case 2: printf (" Enter item : ");
                        scanf ("%d", & item);
        head = dinsert_front (item, head);
        break;
    case 3 : printf (" Enter item : \n");
        scanf ("%d", & item);
        head = insert_leftpos( item, head);
        break;
    case 4 : printf(" Enter item : \n ");
        scanf ("%d", & item);
        head = insert_rightpos (item, head);
        break;
    case 5 : head = ddelete_front (head);
        break;
    case 6 : head = ddelete_rear (head);
        break;
    case 7: printf (" Enter item : ");
        scanf ("%d", & item );
        head = delete_all (item, head);
        break;
    case 8 : printf (" Enter item : ");
        scanf ("%d", & item);
        search (item, head);
        break;
    case 9 : display (head);
        break;
    default : exit (0); } }
return 0; }
```

```
O/P → 1. Insert rear
 2. Insert front
 3. Insert left
 4. Insert right
 ~~6~~5. Delete front
 6. Delete right
 7. Delete dup
8. Search
9. Display
Enter choice : 1
Enter item : 23
```