Doubly_LL.c

Saved

```c
#include<stdio.h>
struct node
 {
   int info;
   struct node *llink;
   struct node *rlink;
   };
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_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 dinsert_rear(int item,NODE head)
{
    NODE temp,cur;
    temp=getnode();
    temp->info=item;
    cur=head->llink;
    head->llink=temp;
    temp->rlink=head;
    temp->llink=cur;
    cur->rlink=temp;
    return head;
}
NODE ddelete_front(NODE head)
{
    NODE cur,next;
    if(head->rlink==head)
    {
        printf("dq empty\n");
        return head;
    }
    cur=head->rlink;
    next=cur->rlink;
    head->rlink=next;
    next->llink=head;
    printf("the node deleted is %d",cur->info);
    freenode(cur);
    return head;
}
NODE ddelete_rear(NODE head)
{
    NODE cur,prev;
    if(head->rlink==head)
    {
        printf("dq empty\n");
        return head;
    }
    cur=head->llink;
    prev=cur->llink;
    head->llink=prev;
    prev->rlink=head;
    printf("the node deleted is %d",cur->info);
    freenode(cur);
    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("Key Not Found\n");
 return head;
}
prev=cur->llink;
printf("Enter item towards Left of %d :",item);
temp=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->llink==head)
{
printf("List Empty\n");
return head;
}
cur=head->llink;
while(cur!=head)
{
if(item==cur->info)break;
cur=cur->llink;
}
if(cur==head)
{
 printf("Key Not Found\n");
 return head;
}
prev=cur->rlink;
printf("Enter item towards Right of %d :",item);
temp=getnode();
scanf("%d",&temp->info);
prev->llink=temp;
temp->rlink=prev;
cur->rlink=temp;
temp->llink=cur;
return head;
}
NODE delete_all_key(int item,NODE head)
{
NODE prev,cur,next;
int count;
    if(head->rlink==head)
    {
     printf("List Empty.");
     return head;
    }
count=0;
cur=head->rlink;
while(cur!=head)
{
  if(item!=cur->info)
  cur=cur->rlink;
  else
 {
   count++;
   prev=cur->llink;
   next=cur->rlink;
   prev->rlink=next;
   next->llink=prev;
   freenode(cur);
   cur=next;
 }
}
if(count==0)
    printf("Key not Found");
else
    printf("Key found in %d Position(s) and are Deleted\n", count);
return head;
}
NODE search(int item,NODE head)
{
NODE prev,cur,next;
int count;
    if(head->rlink==head)
    {
     printf("List Empty.");
     return head;
    }
count=0;
cur=head->rlink;
while(cur!=head)
{
  if(item!=cur->info)
  {
    printf ("Item not Found\n");
    break;
  }
  else
  {
   printf ("Search Successful\n");
   break;
  }
}
}
void display(NODE head)
{
    NODE temp;
    if(head->rlink==head)
    {
        printf("List Empty. Cannot Display Items.\n");
        return;
    }
    printf("Contents of List.\n");
    temp=head->rlink;
    while(temp!=head)
    {
        printf("%d ",temp->info);
        temp=temp->rlink;
    }
    printf("\n");
}
void main()
{
    NODE head,last;
    int item, choice;
    head=getnode();
    head->rlink=head;
    head->llink=head;
    for(;;)
    {
      printf("\n1.Insert Front\n2.Insert Rear\n3.Delete Front\n4.Delete Rear\n5.Insert Left of Node");
      printf("\n6.Insert Right of Node\n7.Delete Duplicates\n8.Search\n9.Display\n10.Exit\n");
      printf("Enter choice :");
      scanf("%d",&choice);
      switch(choice)
      {
          case 1:
              printf("Enter the Item at Front end :");
              scanf("%d",&item);
              last=dinsert_front(item,head);
              break;
          case 2:
              printf("Enter the Item at Rear end :");
              scanf("%d",&item);
              last=dinsert_rear(item,head);
              break;
          case 3:
              last=ddelete_front(head);
              break;
          case 4:
              last=ddelete_rear(head);
              break;
          case 5:
              printf("Enter the Key Item :");
          scanf("%d",&item);
          head=insert_leftpos(item,head);
          break;
          case 6:
              printf("Enter the Key Item :");
          scanf("%d",&item);
          head=insert_rightpos(item,head);
          break;
          case 7:
              printf("Enter the Key Item to be Deleted :");
          scanf("%d",&item);
          delete_all_key(item,head);
          break;
          case 8:
            printf("Enter Item to be Searched :");
          scanf("%d",&item);
          search(item,head);
          break;
          case 9:
              display(head);
              break;
          default:exit(0);
      }
    }
}
```

```
Terminal

1.Insert Front
2.Insert Rear
3.Delete Front
4.Delete Rear
5.Insert Left of Node
6.Insert Right of Node
7.Delete Duplicates
8.Search
9.Display
10.Exit
Enter Choice :2
Enter the Item at Rear end :10

1.Insert Front
2.Insert Rear
3.Delete Front
4.Delete Rear
5.Insert Left of Node
6.Insert Right of Node
7.Delete Duplicates
8.Search
9.Display
10.Exit
Enter Choice :2
Enter the Item at Rear end :20

1.Insert Front
2.Insert Rear
3.Delete Front
4.Delete Rear
5.Insert Left of Node
6.Insert Right of Node
7.Delete Duplicates
8.Search
9.Display
10.Exit
Enter Choice :2
Enter the Item at Rear end :30

1.Insert Front
2.Insert Rear
3.Delete Front
4.Delete Rear
5.Insert Left of Node
6.Insert Right of Node
7.Delete Duplicates
8.Search
9.Display
10.Exit
Enter Choice :5
Enter the Key Item :20
Enter Item towards Left of 20 :19

1.Insert Front
2.Insert Rear
3.Delete Front
4.Delete Rear
5.Insert Left of Node
6.Insert Right of Node
7.Delete Duplicates
8.Search
9.Display
10.Exit
Enter Choice :6
Enter the Key Item :20
Enter Item towards Right of 20 :21

1.Insert Front
2.Insert Rear
3.Delete Front
4.Delete Rear
5.Insert Left of Node
6.Insert Right of Node
7.Delete Duplicates
8.Search
9.Display
10.Exit
Enter Choice :9
Contents of List,
10 19 20 21 30

1.Insert Front
2.Insert Rear
3.Delete Front
4.Delete Rear
5.Insert Left of Node
6.Insert Right of Node
7.Delete Duplicates
8.Search
9.Display
10.Exit
Enter Choice :1
Enter the Item at Front end :19

1.Insert Front
2.Insert Rear
3.Delete Front
4.Delete Rear
5.Insert Left of Node
6.Insert Right of Node
7.Delete Duplicates
8.Search
9.Display
10.Exit
Enter Choice :2
Enter the Item at Rear end :19

1.Insert Front
2.Insert Rear
3.Delete Front
4.Delete Rear
5.Insert Left of Node
6.Insert Right of Node
7.Delete Duplicates
8.Search
9.Display
10.Exit
Enter Choice :9
Contents of List,
19 10 19 20 21 30 19

1.Insert Front
2.Insert Rear
3.Delete Front
4.Delete Rear
5.Insert Left of Node
6.Insert Right of Node
7.Delete Duplicates
8.Search
9.Display
10.Exit
Enter Choice :7
Enter the Key Item to be Deleted:19
Key found in 3 Position(s) and are Deleted

1.Insert Front
2.Insert Rear
3.Delete Front
4.Delete Rear
5.Insert Left of Node
6.Insert Right of Node
7.Delete Duplicates
8.Search
9.Display
10.Exit
Enter Choice :9
Contents of List,
10 20 21 30

1.Insert Front
2.Insert Rear
3.Delete Front
4.Delete Rear
5.Insert Left of Node
6.Insert Right of Node
7.Delete Duplicates
8.Search
9.Display
10.Exit
Enter Choice :8
Enter Item to be Searched :10
Search Successful

1.Insert Front
2.Insert Rear
3.Delete Front
4.Delete Rear
5.Insert Left of Node
6.Insert Right of Node
7.Delete Duplicates
8.Search
9.Display
10.Exit
Enter Choice :8
Enter Item to be Searched :11
Item not Found
```

Stack_Implementation.c

Saved

```c
#include<stdio.h>
struct node
{
  int info;
  struct node *link;
};
typedef struct node *NODE;
NODE getnode()
{
    NODE x;
    x=(NODE)malloc(sizeof(struct node));
    if(x==NULL)
    {
        printf("mem full\n");
        exit(0);
    }
    return x;
}
void freenode(NODE x)
{
    free(x);
}
NODE insert_rear(NODE first,int item)
{
    NODE temp,cur;
    temp=getnode();
    temp->info=item;
    temp->link=NULL;
    if(first==NULL)
        return temp;
    cur=first;
    while(cur->link!=NULL)
    cur=cur->link;
    cur->link=temp;
    return first;
    }
NODE delete_rear(NODE first)
{
NODE cur,prev;
if(first==NULL)
{
printf("stack is empty cannot delete\n");
return first;
}
if(first->link==NULL)
{
printf("item deleted is %d\n",first->info);
free(first);
return NULL;
}
prev=NULL;
cur=first;
while(cur->link!=NULL)
{
prev=cur;
cur=cur->link;
}
printf("item deleted at rear-end is %d\n",cur->info);
free(cur);
prev->link=NULL;
return first;
}
void display(NODE first)
{
 NODE temp;
 if(first==NULL)
 printf("stack empty cannot display items\n");
 for(temp=first;temp!=NULL;temp=temp->link)
  {
  printf("%d\n",temp->info);
  }
}
void main()
{
int item,choice,pos;
NODE first=NULL;
for(;;)
{
printf("\n1.Insert Rear\n2.Delete Rear\n3.Display List\n4.Exit\n");
printf("Enter Choice:");
scanf("%d",&choice);
switch(choice)
 {

  case 1:printf("Enter Item at Rear-end:");
   scanf("%d",&item);
   first=insert_rear(first,item);
   break;
  case 2:first=delete_rear(first);
   break;
  case 3:display(first);
   break;
 default:exit(0);
   break;
 }
}
}
```

Queue_Implementation.c

Saved

```c
#include<stdio.h>
struct node
{
  int info;
  struct node *link;
};
typedef struct node *NODE;
NODE getnode()
{
    NODE x;
    x=(NODE)malloc(sizeof(struct node));
    if(x==NULL)
    {
        printf("Mem Full\n");
        exit(0);
    }
    return x;
}
void freenode(NODE x)
{
    free(x);
}
NODE insert_rear(NODE first,int item)
{
    NODE temp,cur;
    temp=getnode();
    temp->info=item;
    temp->link=NULL;
    if(first==NULL)
        return temp;
    cur=first;
    while(cur->link!=NULL)
        cur=cur->link;
    cur->link=temp;
    return first;
}
NODE delete_front(NODE first)
{
    NODE temp;
    if(first==NULL)
    {
        printf("List is Empty. Cannot Delete!!\n");
        return first;
    }
    temp=first;
    temp=temp->link;
    printf("Item Deleted at Front-end =%d\n",first->info);
    free(first);
    return temp;
}
void display(NODE first)
{
    NODE temp;
    if(first==NULL)
    printf("List Empty. Cannot Display Items.\n");
    for(temp=first;temp!=NULL;temp=temp->link)
    {
        printf("%d\n",temp->info);
    }
}
void main()
{
    int item,choice,pos;
    NODE first=NULL;
    for(;;)
    {
        printf("\n1.Insert Rear\n2.Delete Front\n3.Display List\n4.Exit\n");
        printf("Enter Choice:");
        scanf("%d",&choice);
        switch(choice)
        {
            case 1:
                printf("Enter Item at Rear-end:");
              scanf("%d",&item);
              first=insert_rear(first,item);
              break;
            case 2:
                first=delete_front(first);
              break;
            case 3:
                display(first);
              break;
            default:
                exit(0);
              break;
        }
    }
}
```

```
× Terminal

1.Insert Rear
2.Delete Rear
3.Display List
4.Exit
Enter Choice:1
Enter Item at Rear-end:20

1.Insert Rear
2.Delete Rear
3.Display List
4.Exit
Enter Choice:1
Enter Item at Rear-end:30

1.Insert Rear
2.Delete Rear
3.Display List
4.Exit
Enter Choice:3
10
20
30

1.Insert Rear
2.Delete Rear
3.Display List
4.Exit
Enter Choice:2
item deleted at rear-end is 30

1.Insert Rear
2.Delete Rear
3.Display List
4.Exit
Enter Choice:3
10
20

1.Insert Rear
2.Delete Rear
3.Display List
4.Exit
Enter Choice:4

Process finished.
```

```
× Terminal

1.Insert Rear
2.Delete Front
3.Display List
4.Exit
Enter Choice:1
Enter Item at Rear-end:10

1.Insert Rear
2.Delete Front
3.Display List
4.Exit
Enter Choice:1
Enter Item at Rear-end:30

1.Insert Rear
2.Delete Front
3.Display List
4.Exit
Enter Choice:1
Enter Item at Rear-end:50

1.Insert Rear
2.Delete Front
3.Display List
4.Exit
Enter Choice:3
10
30
50

1.Insert Rear
2.Delete Front
3.Display List
4.Exit
Enter Choice:2
Item Deleted at Front-end =10

1.Insert Rear
2.Delete Front
3.Display List
4.Exit
Enter Choice:3
30
50

1.Insert Rear
2.Delete Front
3.Display List
4.Exit
Enter Choice:4

Process finished.
```

Lab 8 Program: 9/12/20. 1BM19CS090 RAHIL

Implement Stack and Queue using linked list

Code:

```
#include<stdio.h>
struct node
{
  int info;
  struct node *link;
};
typedef struct node *NODE;
NODE getnode()
{
    NODE x;
    x = (NODE) malloc (sizeof (struct node));
    if (x==NULL)
    {
      printf("mem full\n");
      exit(0);
    }
    return x;
}
void freenode (NODE x)
{
    free(x);
}
NODE insert_rear (NODE first, int item)
{
    NODE temp, cur;
    temp = getnode();
    temp -> info = item;
    temp -> link = NULL;
    if (first == NULL)
        return temp;
    cur = first;
    while (cur -> link != NULL)
    cur = cur -> link;
    cur -> link = temp;
    return first;
}
```

```
NODE delete_rear (NODE first)
{
    NODE cur, prev;
    if (first == NULL)
    {
        printf ("Stack is Empty. Cannot Delete\n");
    }   return first;
    if (first -> link == NULL)
    {
        printf (" Item deleted is %d\n", first -> info);
        free (first);
        return NULL;
    }
    prev = NULL;
    cur = first;
    while (cur -> link != NULL)
    {
        prev = cur;
        cur = cur -> link;
    }
    printf (" Item deleted at Rear-end : %d\n ", cur -> info);
    free (cur);
    prev -> link = NULL;
    return first;
}
void display (NODE first)
{
    NODE temp;
    if (first == NULL)
        printf (" Stack Empty. Cannot Delete Items!! "\n");
    for (temp = first ; temp != NULL ; temp = temp -> link)
        printf("%d \n", temp -> info);

id main()

    int item, choice, pos;
    NODE first = NULL ;
```

```
for(;;)
{
    printf("\n1. Insert Rear \n2. Delete Rear \n3. Display List \n4. Exit \n");
    printf("Enter Choice:");
    scanf("%d", &choice);
    switch(choice)
    {
        case 1: printf("Enter Item at Rear-end:");
                scanf("%d", &item);
                first = insert_rear(first, item);
                break;
        case 2: first = delete_rear(first);
                break;
        case 3: display(first);
                break;
        default: exit(0);
                 break;
    }
}
```