Cheatography

Queue Stack Linked List Cheat Sheet

by deo via cheatography.com/190759/cs/39689/

QUEUE

peek()

```
begin procedure peek()
return queue[front]
end procedure

int peek(){
return queue[front]
}
```

QUEUE

isEmpty()

```
begin procedure isEmpty()
if(front<MIN || front>rear)
return true
else
return false
endif
end procedure
OR
bool isEmpty() {
if(front<0||front>rear)
return true
else
return false
```

QUEUE

isFull

```
begin procedure isFull()
if(rear==MAX_SIZE-1)
return true
else
return false
endif
end procedure
OR
bool(isFull()) {
if(rear==MAX_SIZE -1){
return true
else
return false}
```

QUEUE

enqueue

```
procedure enqueue(data)
if queue is full
return overflow
endif
rear<-rear+1
queue[rear]<-data
return true
```

SINGLY LINKED LIST

Insert at front

```
//insertfront(head,x)
{
newnode=create newnode
newnode->data=data
newnode->next=null
if(head!=null)
newnode->next=head
head=newnode
end
```

SINGLY LINKED LIST

Insert at end

```
//insertend(temp,head)
{
newnode->next=null
temp=head
while(temp->next!=null) do
temp=temp->next
end while
temp->next=newnode
newnode->next=null
}
```

SINGLY LINKED LIST

Insert at any position

```
//insertatanypos(head,x,pos)
{
temp=head
while(i<pos)
temp=temp->next
i++
endwhile
newnode=create newnode
newnode->data=data
newnode->next=temp->next
temp->next=newnode
}
```

SINGLY LINKED LIST

Delete from last

```
//deletelast(head,temp,ptr)
{
if(head->next=null)
temp=head
head=null
else
temp=head
while(temp->next!=null)
ptr=temp
```

DOUBLE LINKED LIST

Delete from last

```
if(head==null)
underflow
else
set temp=head
repeat while (temp->next!=null){
temp=temp->next
}
temp->prev->next=null
free(temp)
exit
```

DOUBLE LINKED LIST

Delete from any position

```
if(head==null)
underflow
else
temp=head
repeat while(temp->data=item){
temp=temp->next
}
if(temp->next==null)
return
else
ptr=temp->next
temp->next=ptr->next
ptr->next->prev=temp
free(ptr)
exit
```

SINGLY LINKED LIST

Delete from any position

```
//deleteany(head,x,pos)
{
i=0
temp=head
while(i<pos) {
ptr=temp
temp=temp->next
i++}
ptr->next=temp->next
free(temp)
}
```

SINGLY LINKED LIST

Delete from front

```
//deletefront(head,temp)
{
if(head==null)
no list
```

DOUBLE LINKED LIST

Insert at end

```
temp=head
while(temp!=null)
{
temp=temp->next
}
temp->next=newnode
newnode->prev=temp
newnode->next=null
```

DOUBLE LINKED LIST

Insert at any position

```
temp=head
for(i=0;i<loc;i++)
{
temp=temp->next
if(temp==null)
{
return
}}
ptr->next=temp->next
ptr->prev=temp
temp->next=ptr
ptr->next->prev=ptr
```

CIRCULAR QUEUE

Initialization & Display

```
queue()
begin
front=rear=-1
repeat for i=0 to MAX-1
queue[i]=0
end
AND
Disp()
begin
for i=0 to MAX-1 do
write que[i]
end for
end disp
```

CIRCULAR QUEUE

dequeue

```
begin
if(front==-1)
then write "queue is empty"
else
write "element dequeued is
%d",queue[front]
queue[front]=0
```

```
end procedure
OR
if enqueue(int data)
if(isFull())
return 0;
rear=rear+1;
queue[rear]=data;
return 1;
```

QUEUE

dequeue

```
procedure dequeue
if queue is empty
return underflow
endif
data=queue[front]
front<-front+1
return true
end procedure
OR
int dequeue() {
if(isEmpty())
return 0;
int data;
data=queue[front];
front=front++;
return data;
```

SINGLY LINKED LIST

Initialization

```
struct node
{
int data;
}
struct node *next;
struct node * head = null,
struct node *newnode;
newnode=(struct node*)malloc-
(sizeof(struct node))
newnode->data=data;
newnode->next=null
```

```
temp=temp->next
ptr->next=null
free(temp)
}
```

CIRCULAR QUEUE

enqueue

```
begin
if(front==(rear+1)%MAX)
print queue is full
else read x
if(front==-1)
front=rear=0;
else
rear=(front+1)%MAX
queue[rear]=x
endif
end enqueue
```

DOUBLE LINKED LIST

Delete from front

```
if(head==null)
underflow
set ptr=head
head=head->next
head->prev=null
free(ptr)
exit
```

```
else
temp=head
head=head->next
free(temp)
}
```

DOUBLE LINKED LIST

Initialization

```
struct node
{
struct node *prev
int data
struct node *next
}
struct node *head
```

DOUBLE LINKED LIST

Insert at front

```
if(head==null) do
newnode->next=null
newnode->prev=null
newnode->data=item
head=newnode
else
newnode->next=head
head->prev=newnode
newnode->prev=null
head=newnode
```

```
if(front==rear)
front=rear=-1
else
front=(front+1)%MAX
end dequeue
```