UNIVERSITY PARTNER
UNIVERSITY OF WOLVERHAMPTON

iAcademy
Quality Education Improving Lives

HERALD COLLEGE KATHMANDU

Academic Year	Module	Assessment Number	Assessment Type
S20	Introductory Data Structures and Algorithms (DipIT02)	A1	Assignment Submission

[Assignment Submission]

Student Id : [NP03A190299]
Student Name : [Yogesh Shrestha]
Section : [DC8]
Module Leader : [Mr. Prakash Gautam]

Submitted on : 06-03-2020

Page No.
Date : / /

Tutorial 2

1->. Pseudocodes

Singly link list (SLL)

i) Insert @ Beginning O(1)

```
insertBeg(node)
node.next = head
head = node
size++
```

ii) Insert @ End O(n)

```
temp = head
while (temp.next != 0)
temp = temp.next
temp.next = node
node.next = null
```

iii) Insert @ specific position O(n) size++

```
insertSpf(node)
temp
for i=0 to p-1, do
  temp = temp.next
node.next = temp.next
temp.next = node
size++
```

iv) delete @ beginning. O(1)

```
deleteBeg()
first = first.next
size--
```

v) delete @ End

```
deleteEnd() O(n)
temp = head
while(temp.next!=NULL)
prev = temp
temp = temp.next
prev.next = null
free(temp)
size--
```

vi) delete @ specific position

```
deleteSpf()
temp = head
if i=0 to p-1, do
temp = temp.next
temp.next = temp.next.next
size--
```

Page No.
Date : / /

DLL

a) Insert @ beginning

```
insertB(node)
node.next = head
head.prev = node
head = node
node.prev = null
size++;
```

b) Insert @ End

```
insertEnd(node)
  temp = head
while(temp.next != null)
  temp = temp.next
temp.next = node
node.prev = temp
  node.next = null
  size++
```

c). Insert @ specific position

```
insertSp(node)
  temp;
for i=0 to p-1; do
  temp = temp.next
node.next = temp.next
temp.next.prev = node
node.prev = temp
size++
```

d) Delete @ Beginning

```
deleteBeg()
head = head.next
head.prev = NULL
  size--
```

e) Delete @ End

```
temp = head
while temp.next != NULL
temp = temp.next
temp.prev.next = null
  size--
```

f) Delete @ specific position

```
deleteSp()
for i=0 to p.d:
temp.prev.ned = temp.next
temp.next.prev = temp.prev
size--
```

Page No.
Date : / /

SC

a) insert @ beginning

```
insertBeg(node)
nod.next=head
head:node
last.next=node
size++
```

b) insert @ End

```
insertEnd(node)
last.next=node
node.next=first
size++
```

b) insert @ specific position

```
insertSp(node)
for i=0 to p-1, do.
temp = temp.next
node.next = temp.next
temp.next = node
size++
```

d) delete @ start/Begining

```
delete()
head = head.next
last.next = head
size--
```

e) delete @ End

```
deleteEnd()
while(temp.next != last)
temp = temp.next
temp.next = first;
last = temp
size--
```

f) delete @ specific position

```
deleteSp()
for i=0 to p-1; do:
temp = temp.next
temp.next = temp.next.next
size--
```

Page No.
Date : / /

DC

a) Insert @ Beginning

```
insertBeg(node)
node.next = first
first.prev = node
node.prev = last
last.next = node
first = node
size++
```

b) Insert @ End.

```
insertEnd(node)
last.next = node
node.prev = last
first.prev = node
node.next = first
size++
```

c) Insert @ specific position

```
insertSpl(node)
for i=0 to p-1; do
temp = temp.next
node.next = temp.next
temp.next.prev = node
temp.next = node
node.prev = temp
size++
```

d) delete @ beginning

```
first = first.next
first.prev = last
last.next = first
size--
```

e) delete @ End

```
deleteEnd()
last.prev.next = first
last = last.prev
first.prev = last
size--
```

f) delete @ specific position

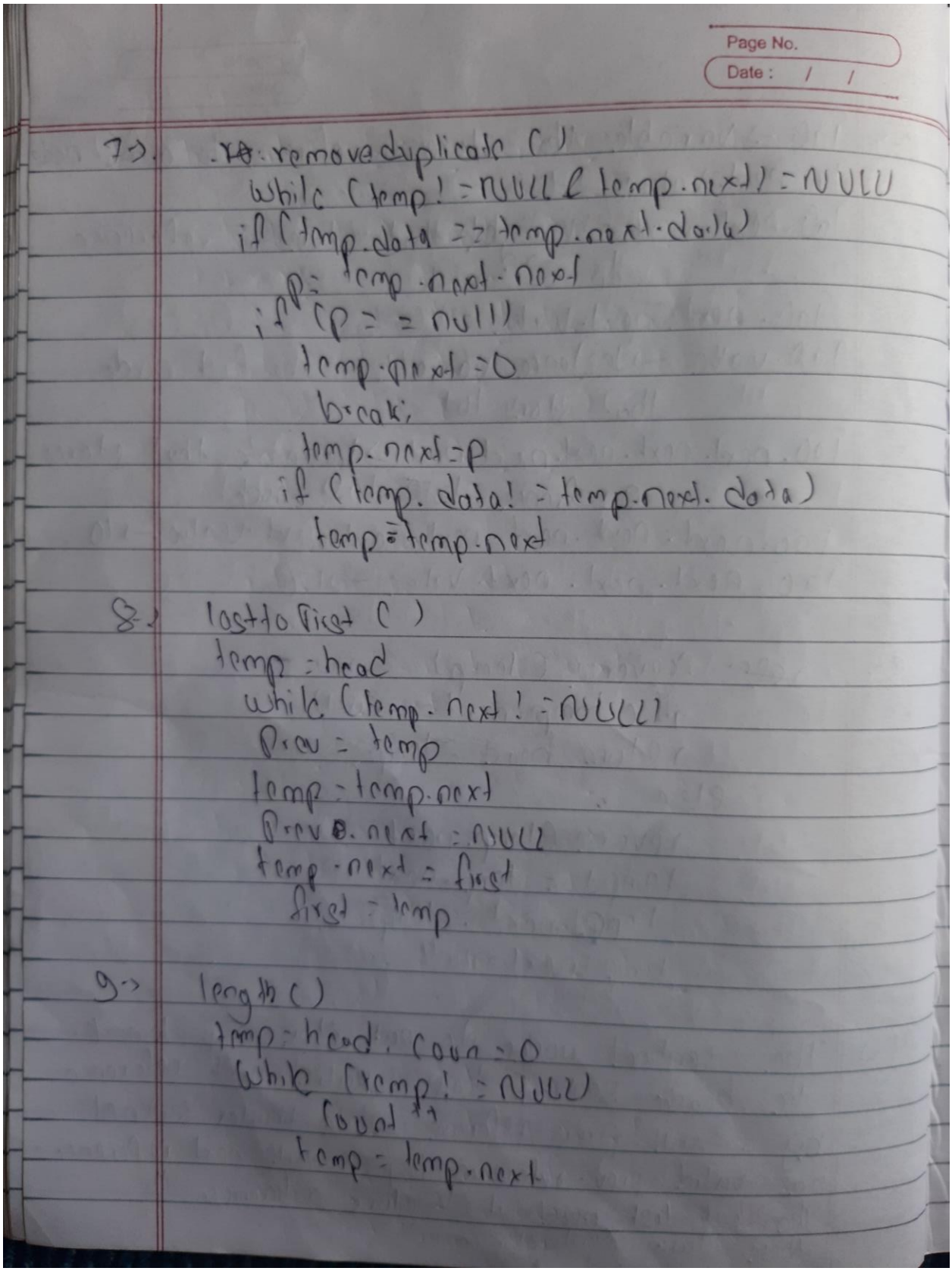
```
deleteSpl()
for i=0 to p; do:
temp = temp.next
temp.next.prev = temp.prev
size--
```

Page No.
Date : / /

2-> Tao -> variable that stores a reference to a first node
Tao next. value -> 109
Tao. next. next -> variable that stores a reference to 3rd node.
Tao. next. next. value -> 110
Tao. value -> Instance variable in the first node that stores 101
Tao. next. next. next. next. next -> variable that stores a refrence to a last node.
Tao. next. next. next. next. next. next. value -> 10.
Tao. next. next. next. value -> 120.

3-> ~~refe~~ reverse (temp)
if (temp. next == null)
return head = temp,
else
reverse (temp. next);
temp1 = temp. next
temp1. next = temp
temp. next = null

4-> The sentinel nodes do not store any elements.
The header sentinel has a valid next reference by a null prev reference, while trailer sentinel has valid prev. reference by a null next reference.
The link list need it to store references to these two sentinel.



Page No.
Date : / /

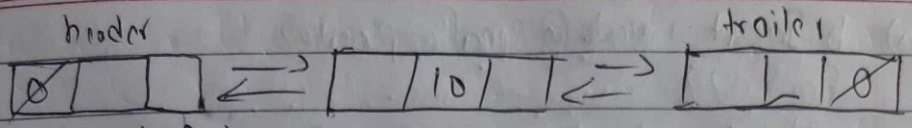
7=>
```
removeduplicate ()
while (temp != NULL && temp.next != NULL)
if (tmp.data == temp.next.data)
    p = temp.next.next
    if (p == null)
        temp.next = 0
        break;
    temp.next = p
    if (temp.data != temp.next.data)
        temp = temp.next
```

8.)
```
lasttofirst ()
temp = head
while (temp.next != NULL)
    prev = temp
    temp = temp.next
    prev.next = NULL
    temp.next = first
    first = temp
```

9->
```
length ()
tmp = head; coun = 0
while (temp != NULL)
        count++
    temp = temp.next
```

Page No.
Date : / /

5→



insert ( )

header node x=10

node.next = trailer

trailer.prev = node

header.next = node

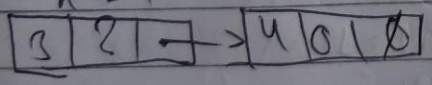
node.prev = header

node.value = x

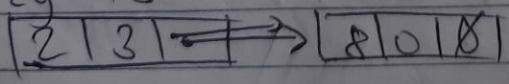
Size ++

6→ The biggest +ve integer that we can store in a variable of the type int is $2^{31} - 1$ as 32-bit 2^{32} ⇒ here are both +ve & -ve type $\frac{2^{32}}{2}$ 2^{31} do -1 as there include 0 also so $2^{31} - 1$

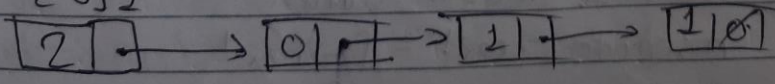
a) 3 n 2 4



b) 2 y 3 18



c) 2 0 1 1



Page No.
Date : / /

7 a) b Insert a node @ Sprific position

for SLL

```
Struct node *newnode, *p
int x, a, i=1;
Printf ("Enter the location index)
Scanf ("%d", &x);
  printf ("data")
  Scanf ("%d", &a);
  whil (i<x){
    p = p->next;
  i++; }
  new node -> data = a;
   newnode->next = p->next)
   p->next = newnode
```

for DLL

```
Struct node * newnode, *p
  int location, data, i=1;
   whil (i<location)
      p = p->next;
       i++;
    }
  newnode->data = data;
  newnode->next = p->next)
  Newnode->previous = p;
  newnode->next->Previous = newnode
   p->next = newnode;
```

Page No.
Date : / /

b) Delete a node @ specific position

SLL

```
struct node temp, *p
 int location, i=1;
 while (i<a-1)
 {
  p=p->next;
  i++;
 }
 temp=p->next;
 p->next=temp->next;
 temp->next=NULL;
 free(temp);
```

DLL

```
 Struct node *temp, *p;
  int location, i=1;
  While (i<a)
  {
   p= p->next;
   i++;
  }
 p->previous->next = p->next;
 p->next->previous = p->previous
  free (p);
```

Page No.
Date : / /

2 -> Program to reverse a Single linked list, recursively.

```
Void reverse (struct node **p)
{
 Struct node *prev = NULL;
 Struct node *current = *p;
  Struct node *next = NULL;
 while (current != NULL){
  next = current->next;
  current->next = prev;
    prev = current;
    current = next;
    }
    *p = prev;
  }
```

4 : Program to move last node to front in linked list.

```
  last to front ()
 Struct node *p, *temp, *seclast;
  p = temp = Head;
 while (p->next != NULL)
 {
    seclast = p;
    p = p->next;
  }
 p->next = temp;
 Seclast->next = NULL;  Head = p;
```

Page No.
Date : / /

5. My solution in C to calculate the total length of a linked list.

```
length ():
    struct node *P;
    P = A Head;
    while (P != NULL ):
        length ++;
        P = P -> next
    }
```