Assignment - 5

Q1. Write pseudocode for singly linked list operations.

#include <stdio.h>

#include <malloc.h>

#include <stdlib.h>

struct node {

int value;

struct node \*next;

};

void insert();

void display();

void delete();

int count();

typedef struct node DATA\_NODE;

DATA\_NODE \*head\_node, \*first\_node, \*temp\_node = 0, \*prev\_node, next\_node;

int data;

int main() {

int option = 0;

while (option < 5) {

printf("\nOptions\n");

printf("1 : Insert into Linked List \n");

printf("2 : Delete from Linked List \n");

printf("3 : Display Linked List\n");

printf("4 : Count Linked List\n");

printf("Others : Exit()\n");

printf("Enter your option:");

scanf("%d", &option);

switch (option) {

case 1:

insert();

break;

case 2:

delete();

break;

case 3:

display();

break;

case 4:

count();

break;

default:

break;

}

}

return 0;

}

void insert() {

printf("\nEnter Element for Insert Linked List : \n");

scanf("%d", &data);

temp\_node = (DATA\_NODE \*) malloc(sizeof (DATA\_NODE));

temp\_node->value = data;

if (first\_node == 0) {

first\_node = temp\_node;

} else {

head\_node->next = temp\_node;

}

temp\_node->next = 0;

head\_node = temp\_node;

fflush(stdin);

}

void delete() {

int countvalue, pos, i = 0;

countvalue = count();

temp\_node = first\_node;

printf("\nDisplay Linked List : \n");

printf("\nEnter Position for Delete Element : \n");

scanf("%d", &pos);

if (pos > 0 && pos <= countvalue) {

if (pos == 1) {

temp\_node = temp\_node -> next;

first\_node = temp\_node;

printf("\nDeleted Successfully \n\n");

} else {

while (temp\_node != 0) {

if (i == (pos - 1)) {

prev\_node->next = temp\_node->next;

if(i == (countvalue - 1))

{

head\_node = prev\_node;

}

printf("\nDeleted Successfully \n\n");

break;

} else {

i++;

prev\_node = temp\_node;

temp\_node = temp\_node -> next;

}

}

}

} else

printf("\nInvalid Position \n\n");

}

void display() {

int count = 0;

temp\_node = first\_node;

printf("\nDisplay Linked List : \n");

while (temp\_node != 0) {

printf("# %d # ", temp\_node->value);

count++;

temp\_node = temp\_node -> next;

}

printf("\nNo Of Items In Linked List : %d\n", count);

}

int count() {

int count = 0;

temp\_node = first\_node;

while (temp\_node != 0) {

count++;

temp\_node = temp\_node -> next;

}

printf("\nNo Of Items In Linked List : %d\n", count);

return count;

}

Q2. Write pseudocode for stack using singly linked list.

Step 1 - Include all the header files which are used in the program. And declare all the

user defined functions.

Step 2 - De1ne a 'Node' structure with two members data and next.

Step 3 - De1ne a Node pointer 'top' and set it to NULL.

Step 4 - Implement the main method by displaying a Menu with a list of operations

and make suitable function calls in the main method.

push(value) - Inserting an element into the Stack

insert a new node

Step 1 - Create a newNode with a given value.

Step 2 - Check whether stack is Empty (top == NULL)

Step 3 - If it is Empty, then set newNode â next = NULL.

Step 4 - If it is Not Empty, then set newNode â next = top.

Step 5 - Finally, set top = newNode.

pop() - Deleting an Element from a Stack

delete a node

Step 1 - Check whether the stack is Empty (top == NULL).

Step 2 - If it is Empty, then display "Stack is Empty!"

and terminate the function

Step 3 - If it is Not Empty, then de1ne a Node pointer 'temp' and set it to 'top'.

Step 4 - Then set 'top = top â next'.

Step 5 - Finally, delete 'temp'. (free(temp)).

display() - Displaying stack of elements

display the elements(nodes)

Step 1 - Check whether the stack is Empty (top == NULL).

Step 2 - If it is Empty, then display 'Stack is Empty!!!' and terminate the function.

Step 3 - If it is Not Empty, then de1ne a Node pointer 'temp' and initialize with top.

Step 4 - Display 'temp â data --->' and move it to the next node. Repeat the same

until temp reaches the 1rst node in the stack. (temp â next != NULL).

Step 5 - Finally! Display 'temp â data ---> NULL'.

Q3. Pseudocode for queue using Linked List.

Step 1 - Include all the header 1les which are used in the program. And declare all the

user defined functions.

Step 2 - De1ne a 'Node' structure with two members data and next.

Step 3 - De1ne two Node pointers 'front' and 'rear' and set both to NULL.

Step 4 - Implement the main method by displaying a Menu of list of operations and

make suitable function calls in the main method to perform user selected operation.

enQueue(value) - Inserting an element into the Queue

Step 1 - Create a newNode with a given value and set 'newNode â next' to NULL.

Step 2 - Check whether queue is Empty (rear == NULL)

Step 3 - If it is Empty then, set front = newNode and rear = newNode.

Step 4 - If it is Not Empty then, set rear â next = newNode and rear = newNode.

deQueue() - Deleting an Element from Queue

Step 1 - Check whether the queue is Empty (front == NULL).

Step 2 - If it is Empty, then display "Queue is Empty!!! Deletion is not possible!!!" and

terminate from the function

Step 3 - If it is Not Empty then, de1ne a Node pointer 'temp' and set it to 'front'.

Step 4 - Then set 'front = front â next' and delete 'temp' (free(temp)).

display() - Displaying the elements of Queue

Step 1 - Check whether the queue is Empty (front == NULL).

Step 2 - If it is Empty then, display 'Queue is Empty!!!' and terminate the function.

Step 3 - If it is Not Empty then, de1ne a Node pointer 'temp' and initialize with front.

Step 4 - Display 'temp â data --->' and move it to the next node. Repeat the same until

'temp' reaches to 'rear' (temp â next != NULL).

Step 5 - Finally! Display 'temp â data ---> NULL'.

**Q4.** Pseudocode for singly circular linked list operations:

(1) Insert ﬁrst

(2) Insert last

(3) Insert after

(4) Delete ﬁrst

(4) Delete last

(6) Delete after

Step 1 - Include all the header 1les which are used in the program.

Step 2 - Declare all the user de1ned functions.

Step 3 - Define a Node structure with two members data and next

Step 4 - Define a Node pointer 'head' and set it to NULL.

Step 5 - Implement the main method by displaying the operations menu and make

suitable function calls in the main method to perform user selected operation.

Inserting At Beginning of the list

Step 1 - Create a newNode with a given value.

Step 2 - Check whether list is Empty (head == NULL)

Step 3 - If it is Empty then, set head = newNode and newNode->next = head .

Step 4 - If it is Not Empty then, de1ne a Node pointer 'temp' and initialize with 'head'.

Step 5 - Keep moving the 'temp' to its next node until it reaches the last node (until

'temp -> next == head').

Step 6 - Set 'newNode -> next =head', 'head = newNode' and 'temp -> next = head'.

Inserting At End of the list

Step 1 - Create a newNode with a given value.

Step 2 - Check whether the list is Empty (head == NULL).

Step 3 - If it is Empty then, set head = newNode and newNode -> next = head.

Step 4 - If it is Not Empty then, de1ne a node pointer temp and initialize with head.

Step 5 - Keep moving the temp to its next node until it reaches the last node in the list

(until temp â next == head).

Step 6 - Set temp â next = newNode and newNode -> next = head.

Inserting At Specific location in the list (After a Node)

Step 1 - Create a newNode with a given value.

Step 2 - Check whether list is Empty (head == NULL)

Step 3 - If it is Empty then, set head = newNode and newNode â next = head.

Step 4 - If it is Not Empty then, de1ne a node pointer temp and initialize with head.

Step 5 - Keep moving the temp to its next node until it reaches the node after which we

want to insert the newNode (until temp1 â data is equal to location, here location is

the node value after which we want to insert the newNode).

Step 6 - Every time check whether temp is reached to the last node or not. If it is reached

to the last node then display 'Given node is not found in the list!!! Insertion is not

possible!!!' and terminate the function. Otherwise move the temp to the next node.

Step 7 - If temp is reached to the exact node after which we want to insert the newNode

then check whether it is the last node (temp â next == head).

Step 8 - If temp is the last node then set temp â next = newNode and newNode â

next = head.

Step 8 - If temp is not the last node then set newNode â next = temp â next and

temp â next = newNode.

Deleting from Beginning of the list

Step 1 - Check whether list is Empty (head == NULL)

Step 2 - If it is Empty then, display 'List is Empty!!! Deletion is not possible' and

terminates the function.

Step 3 - If it is Not Empty then, de1ne two Node pointers 'temp1' and 'temp2' and

initialize both 'temp1' and 'temp2' with head.

Step 4 - Check whether list is having only one node (temp1 â next == head)

Step 5 - If it is TRUE then set head = NULL and delete temp1 (Setting Empty list

conditions)

Step 6 - If it is FALSE move the temp1 until it reaches the last node. (until temp1 â

next == head )

Step 7 - Then set head = temp2 â next, temp1 â next = head and delete temp2.

Deleting from End of the list

Step 1 - Check whether list is Empty (head == NULL)

Step 2 - If it is Empty then, display 'List is Empty!!! Deletion is not possible' and

terminates the function.

Step 3 - If it is Not Empty then, de1ne two Node pointers 'temp1' and 'temp2' and

initialize 'temp1' with head.

Step 4 - Check whether list has only one Node (temp1 â next == head)

Step 5 - If it is TRUE. Then, set head = NULL and delete temp1. And terminate from

the function. (Setting Empty list condition)

Step 6 - If it is FALSE. Then, set 'temp2 = temp1 ' and move temp1 to its next node.

Repeat the same until temp1 reaches the last node in the list. (until temp1 â next ==

head)

Step 7 - Set temp2 â next = head and delete temp1.

Deleting a Speci1c Node from the list

Step 1 - Check whether list is Empty (head == NULL)

Step 2 - If it is Empty then, display 'List is Empty!!! Deletion is not possible' and

terminates the function.

Step 3 - If it is Not Empty then, de1ne two Node pointers 'temp1' and 'temp2' and

initialize 'temp1' with head.

Step 4 - Keep moving the temp1 until it reaches the exact node to be deleted or to the

last node. And every time set 'temp2 = temp1' before moving the 'temp1' to its next

node.

Step 5 - If it is reached to the last node then display 'Given node not found in the list!

Deletion is not possible!!!'. And terminate the function.

Step 6 - If it is reached to the exact node which we want to delete, then check whether

list is having only one node (temp1 â next == head)

Step 7 - If the list has only one node and that is the node to be deleted then set head =

NULL and delete temp1 (free(temp1)).

Step 8 - If the list contains multiple nodes then check whether temp1 is the 1rst node in

the list (temp1 == head).

Step 9 - If temp1 is the 1rst node then set temp2 = head and keep moving temp2 to its

next node until temp2 reaches the last node. Then set head = head â next, temp2 â

next = head and delete temp1.

Step 10 - If temp1 is not the 1rst node then check whether it is the last node in the list

(temp1 â next == head).

Step 11- If temp1 is the last node then set temp2 â next = head and delete temp1

(free(temp1)).

Step 12 - If temp1 is not the 1rst node and not the last node then set temp2 â next =

temp1 â next and delete temp1 (free(temp1)).

Displaying a circular Linked List

Step 1 - Check whether list is Empty (head == NULL)

Step 2 - If it is Empty, then display 'List is Empty!!!' and terminate the function.

Step 3 - If it is Not Empty then, de1ne a Node pointer 'temp' and initialize with head.

Step 4 - Keep displaying temp â data with an arrow (--->) until temp reaches to the last

node

Step 5 - Finally display temp â data with an arrow pointing to head â data.

**Q5.** Pseudocode for doubly circular linked list operations:

(1) Insert ﬁrst

(2) Insert last

(3) Delete ﬁrst

(4) Delete last

Insertion at the Beginning

Step 1:

IF AVAIL = NULL

Write OVERFLOW

Go to Step 13

[END OF IF]

Step 2: SET NEW\_NODE = AVAIL

Step 3: SET AVAIL = AVAIL -> NEXT

Step 4: SET NEW\_NODE -> DATA = VAL

Step 5: SET PTR = START

Step 6: Repeat Step 7 while PTR -> NEXT!= START

Step 7:

SET PTR = PTR -> NEXT

[END OF LOOP]

Step 8: SET PTR -> NEXT = NEW\_NODE

Step 9: SET NEW\_NODE -> PREV = PTR

Step 10: SET NEW\_NODE -> NEXT = START

Step 11: SET START -> PREV = NEW\_NODE

Step 12: SET START = NEW\_NODE

Step 13: EXIT

Insertion at the End

Step 1:

IF AVAIL = NULL

Write OVERFLOW

Go to Step 12

[END OF IF]

Step 2: SET NEW\_NODE = AVAIL

Step 3: SET AVAIL = AVAIL -> NEXT

Step 4: SET NEW\_NODE -> DATA = VAL

Step 5: SET NEW\_NODE > NEXT = START

Step 6: SET PTR = START

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

Step 8:

SET PTR = PTR -> NEXT

[END OF LOOP]

Step 9: SET PTR -> NEXT = NEW\_NODE

Step 10: SET NEW\_NODE -> PREV = PTR

Step 11: SET START -> PREV = NEW\_NODE

Step 12: EXIT

Delete first - Algorithm

Step 1:

IF START = NULL

Write UNDERFLOW

Go to Step 8

[END OF IF]

Step 2: SET PTR = START

Step 3: Repeat Step 4 while PTR -> NEXT != START

Step 4:

SET PTR = PTR -> NEXT

[END OF LOOP]

Step 5: SET PTR -> NEXT = START -> NEXT

Step 6: SET START -> NEXT -> PREV = PTR

Step 7: FREE START

Step 8: SET START = PTR -> NEXT

Delete end - Algorithm

Step 1:

IF START = NULL

Write UNDERFLOW

Go to Step 8

[END OF IF]

Step 2: SET PTR = START

Step 3: Repeat Step 4 while PTR -> NEXT != START

Step 4:

SET PTR = PTR -> NEXT

[END OF LOOP]

Step 5: SET PTR -> PREV -> NEXT = START

Step 6: SET START -> PREV = PTR -> PREV

Step 7: FREE PTR

Step 8: EXIT