DATA STRUCTURE AND ALGORITHM

# ASSIGNMENT 1

Doubly Linked List, Circular Linked List, Finding Complexity

Alok singh

18102207

A8

**Ques 1 and 2--** #include <stdio.h> #include <iostream> #include <stdlib.h> using namespace std; **struct Node{**

**int data;**

**struct Node \*next, \*prev;**

**};**

**void push(struct Node\*\* head, int key)**{

struct Node\* node = (struct Node\*)malloc(sizeof(struct Node)); node->data = key;

node->prev = nullptr; node->next = \*head;

if (\*head != nullptr) { (\*head)->prev = node;

}

\*head = node;

}

**void print\_list(struct Node\* head)**{ struct Node\* temp = head;

while (temp!= nullptr)

{

cout<<temp->data<<"\t"; temp = temp->next;

}

cout<<endl;

}

**void split(struct Node\* head, struct Node\*\* a, struct Node\*\* b)**{ struct Node\* slow = head;

struct Node\* fast = head->next; while (fast != nullptr)

{

fast = fast->next; if (fast != nullptr)

{

slow = slow->next; fast = fast->next;

}

}

\*b = slow->next; slow->next = nullptr;

}

**struct Node\* merge(struct Node\* a, struct Node\* b)**{ if (a == nullptr) {

return b;

}

if (b == nullptr) { return a;

}

if (a->data <= b->data){

a->next = merge(a->next, b); a->next->prev = a;

a->prev = nullptr; return a;

} else {

b->next = merge(a, b->next); b->next->prev = b;

b->prev = nullptr; return b;

}

}

**void mergesort(struct Node\*\* head)**{ struct Node\* temp = \*head;

if (\*head == nullptr || (\*head)->next == nullptr) { return;

}

struct Node\* a = \*head, \*b = NULL; split(\*head, &a, &b); mergesort(&a);

mergesort(&b);

\*head = merge(a, b);

}

**void insert\_at\_loc(struct Node\* head, int el, int pos)**{ struct Node\* temp = head;

struct Node\* node = (struct Node\*)malloc(sizeof(struct Node)); node->data = el;

for(int i=1;i<pos - 1;i++){ temp = temp->next;

}

node->prev = temp;

node->next = temp->next; temp->next = node;

temp->next->prev = node;

}

int main(void)

{

struct Node\* head = nullptr; push(&head, 24); push(&head,13); push(&head,56); push(&head,39); push(&head,16); push(&head,42); push(&head,32); cout<<"Your linked list \n";

print\_list(head);

cout<<"Your linked list after sorting \n";

mergesort(&head); print\_list(head);

int val, loc;

cout<<"Enter an element and it's location: "; cin>>val>>loc;

insert\_at\_loc(head, val, loc);

cout<<"Modified linked list-- \n"; print\_list(head);

return 0;

}

**Ques 3 and 4---** #include <iostream> using namespace std; **struct clist{**

**int data;**

**struct clist\* next;**

**};**

**void insert\_cl(struct clist\*\* head1, int new\_data)**{ struct clist\* temp = \*head1;

struct clist\* new\_node = (struct clist\*)malloc(sizeof(struct clist)); new\_node->data = new\_data;

if(\*head1 == nullptr){

\*head1 = new\_node; new\_node->next = \*head1;

}else{

while(temp->next!= \*head1){ temp = temp->next;

}

temp->next = new\_node; new\_node->next = \*head1;

}

}

**int length(struct clist\* head)**{ int lsize=1;

if(head == nullptr){ return lsize;

}else{

struct clist\* temp = head; while(temp->next!=head){

lsize++; temp=temp->next;

}

}

return lsize;

}

**void display(struct clist\* head)**{ struct clist\* temp = head;

if(head == nullptr) cout<<"Linked list is empty";

else{

while(temp->next!= head){ cout<<temp->data<<"\t"; temp = temp->next;

}

cout<<temp->data<<endl;

}

}

**struct clist \*merge\_list(struct clist\* head1, struct clist\* head2)**{ struct clist\* temp = nullptr;

struct clist\* res\_list=nullptr; res\_list = head1;

temp = head1;

while(temp->next != head1){ temp = temp->next;

}

temp->next = head2; temp = head2;

while(temp->next != head2){ temp = temp->next;

}

temp->next = res\_list; return(res\_list);

}

int main()

{

struct clist\* head1 = nullptr; struct clist\* head2 = nullptr; struct clist\* head3 = nullptr; cout<<"enter length of first list: "; int n1;

cin>>n1;

for(int i=1; i<n1;i++){ cout<<"\n enter val: "; int val;

cin>>val; insert\_cl(&head1,val);

}

cout<<"\n enter length of second list: "; int n2;

cin>>n2;

for(int i=1; i<n2;i++){ cout<<"\n enter val: "; int val;

cin>>val; insert\_cl(&head2,val);

}

display(head1); display(head2);

head3 = merge\_list(head1, head2); cout<<"New linked list \n";

display(head3); int len;

len = length(head3);

cout<<"\n"<<len; return 0;

}

# Ques 5---

1. **O(n)**

# O(n)

1. **O(n)**