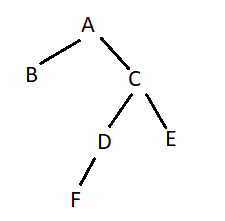
1. 

int f(node\* r)

{

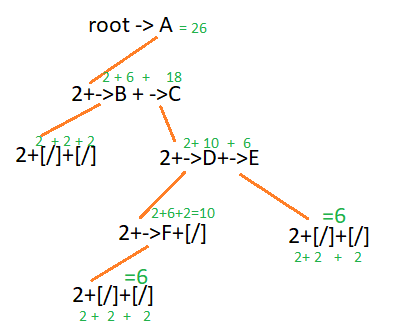
if (r==NULL) return 2;

else return 2+f(r->left)+f(r->right);

}

calling statement: cout<<f(root);

ANSWER:

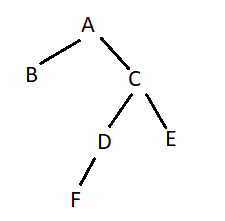


1. Apply inorder and postorder to display all nodes

ANSWER:

inorder: B A F D C E

post: B F D E C A



1. Write a recursive function to display ONLY the right-subtree

ANSWER:

display(root->right)

void display(node \*r)

{

//use any traversal algorithm

if(r!=NULL)

display(r->left)

display(r->right)

cout<<p->info;

}

1. Find the big-OH of the following
2. f(n) = (n+2)^3 (n-1)^4 + n^6 log(n^2 + 2)

ANSWER:

O(n^3) O(n^4) + O(n^6) O(log n)

O(n^7) + O(n^6 log n)

O(max(n^7, n^6 log n))

=O(n^7)

1. int f(int n)

{

if (n<=2) return 0;

else return f(n-2)+f(n-1);

}

answer: O(2^n)

1. Given: List->[ |]->[ |]. . . .[ |/]

and declaration:

struct node

{

int info; node \*next;

};

1. write statement to convert the linked list to a circular LL.

ANSWER:

node \*p = list;

while(p->next != NULL)

{p=p->next;}

p->next=list;

1. write statement to count the number of nodes in LL.

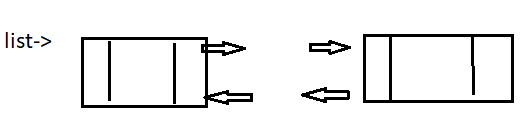
ANSWER:

int total=0, p=list;

do{p=p->next;

total ++;}while(p!=list);

1. given



and declaration

struct node

{

int info;

node \*next;

node \*back;

};

write statement to

1. delete the first node

ANSWER:

node \*p = list;

list = p->next;

//p->next->back = list; don’t need this apparently

list->back = NULL;

delete(p);

1. insert a new node with info 100 at the rear of the list

ANSWER:

node \*r = new node;

r->info = 100;

r->next=NULL;

node \*p = list;

while(p->next!=NULL)

{p=p->next;}

p->next=r; r->back=p;

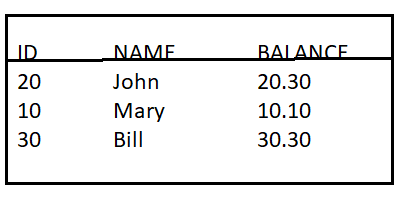
1. write a recursive function

to compute

5^2+5^3+. . . . +5^10

1. what is a big-Oh of searching for item x in a
2. linked list with n nodes
3. BST with n nodes
4. use the following class declaration modify the class to store the following data in each node

file.txt



1. class BST

{

private:

struct node{

int info;

node\* right;

node \*left;

};

node \*root;

public:

BST(){root=NULL;}

void insert(node \* &r, int x)

{

node \*p = new node;

p->info = x; p->left=NULL; p->right=NULL;

if(r==NULL){r=p;}

else

{

if(x<r->info) insert(r->left, x);

if(x>r->info) insert(r->right, x)

}//endelse

}//end insert

void display(node \*r)

{

if(r!=NULL)

{

display(r->left);

cout<<r->info;

display(r->left);

}//endif

}//end display

//overridden functions

void insert(int x){insert(root, x);}

void display() {display(root);}

}; //end class BST