1. What is the difference between win 8 and win 7

* No Start Button – Metro UI
* Simpler Task Manager
* Fast Startup Mode
* Plug-in Free Browsing
* Refresh/Reset Your PC

1. What is the diff between thread and process

* Threads are easier to create than processes since they

don't require a separate address space.

* Multithreading requires careful programming since threads

share data strucures that should only be modified by one thread

at a time. Unlike threads, processes don't share the same

address space.

* Threads are considered lightweight because they use far

less resources than processes.

* Processes are independent of each other. Threads, since they

share the same address space are interdependent, so caution

must be taken so that different threads don't step on each other.

* A process can consist of multiple threads.

1. When u open a new tab in browser, the new tab is a thread or process(questions related)

Puzzles:

1. Cut round cake into 8 equal pieces u can use knife only 3 times
2. 8 marbles puzzle(very famous)
3. Given a crescent moon shape. Cut it into 7 parts with 2 cuts at max.
4. Program to print last nth element of linked list in single traversal

* Find out the length of the list. And then traverse to len – n+1 node from beginning.

/\* Function to get the nth node from the last of a linked list\*/

void printNthFromLast(struct node\* head, int n)

{

int len = 0, i;

struct node \*temp = head;

// 1) count the number of nodes in Linked List

while (temp != NULL)

{

temp = temp->next;

len++;

}

// check if value of n is not more than length of the linked list

if (len < n)

return;

temp = head;

// 2) get the (n-len+1)th node from the begining

for (i = 1; i < len-n+1; i++)

temp = temp->next;

printf ("%d", temp->data);

return;

}

* Maintain two pointers – reference pointer and main pointer. Initialize both reference and main pointers to head. First move reference pointer to n nodes from head. Now move both pointers one by one until reference pointer reaches end. Now main pointer will point to nth node from the end. Return main pointer.

/\* Function to get the nth node from the last of a linked list\*/

void printNthFromLast(struct node \*head, int n)

{

struct node \*main\_ptr = head;

struct node \*ref\_ptr = head;

int count = 0;

if(head != NULL)

{

while( count < n )

{

if(ref\_ptr == NULL)

{

printf("%d is greater than the no. of "

"nodes in list", n);

return;

}

ref\_ptr = ref\_ptr->next;

count++;

} /\* End of while\*/

while(ref\_ptr != NULL)

{

main\_ptr = main\_ptr->next;

ref\_ptr = ref\_ptr->next;

}

printf("Node no. %d from last is %d ",

n, main\_ptr->data);

}

}

1. Program to create mirror image of binary tree

(1) Call Mirror for left-subtree i.e., Mirror(left-subtree)

(2) Call Mirror for right-subtree i.e., Mirror(left-subtree)

(3) Swap left and right subtrees.

temp = left-subtree

left-subtree = right-subtree

right-subtree = temp

#include<stdio.h>

#include<stdlib.h>

/\* A binary tree node has data, pointer to left child

   and a pointer to right child \*/

struct node

{

    int data;

    struct node\* left;

    struct node\* right;

};

/\* Helper function that allocates a new node with the

   given data and NULL left and right pointers. \*/

struct node\* newNode(int data)

{

  struct node\* node = (struct node\*)

                       malloc(sizeof(struct node));

  node->data = data;

  node->left = NULL;

  node->right = NULL;

  return(node);

}

/\* Change a tree so that the roles of the  left and

    right pointers are swapped at every node.

 So the tree...

       4

      / \

     2   5

    / \

   1   3

 is changed to...

       4

      / \

     5   2

        / \

       3   1

\*/

void mirror(struct node\* node)

{

  if (node==NULL)

    return;

  else

  {

    struct node\* temp;

    /\* do the subtrees \*/

    mirror(node->left);

    mirror(node->right);

    /\* swap the pointers in this node \*/

    temp        = node->left;

    node->left  = node->right;

    node->right = temp;

  }

}

/\* Helper function to test mirror(). Given a binary

   search tree, print out its data elements in

   increasing sorted order.\*/

void inOrder(struct node\* node)

{

  if (node == NULL)

    return;

  inOrder(node->left);

  printf("%d ", node->data);

  inOrder(node->right);

}

/\* Driver program to test mirror() \*/

int main()

{

  struct node \*root = newNode(1);

  root->left        = newNode(2);

  root->right       = newNode(3);

  root->left->left  = newNode(4);

  root->left->right = newNode(5);

  /\* Print inorder traversal of the input tree \*/

  printf("\n Inorder traversal of the constructed tree is \n");

  inOrder(root);

  /\* Convert tree to its mirror \*/

  mirror(root);

  /\* Print inorder traversal of the mirror tree \*/

  printf("\n Inorder traversal of the mirror tree is \n");

  inOrder(root);

  getchar();

  return 0;

}

1. 25 horses puzzles(famous)
2. Hour glass puzzle(famous)