Karadeniz Technical University
Department of Computer Engineering
Lecturer Omer CAKIR

COM 205 Data Structures
Resit Exam, 01.22.2015, 13:00, D-1
Time : **75** Minutes

NUMBER : ..... **NAME :**	**EXAM GRADE**	
Rules to be Obeyed During the Exam **SIGNATURE :**	[.....]	.....

1. Cell phones are not allowed to be used as a calculator or a watch. They must be switched off and placed in the pocket.
2. Brief information about the exam will be given at the begining, then no one is not allowed to ask a question during the exam.
3. Do not to forget to sign this paper after writing your number and name.

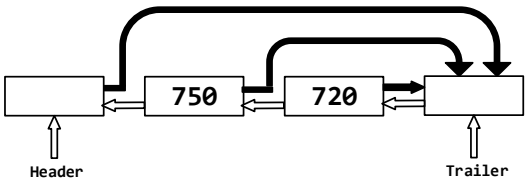
```
void bitOrder(Node* v)
{
	if (v->right != NULL)
	{
		cout << v->elt << " ";
		bitOrder(v->right);
	}
	else
		cout << v->elt << " ";

	if (v->left != NULL)
		bitOrder(v->left);
}

void main()
{
	LinkedBinaryTree Tree;

	Tree.addRoot();
	Tree.root->elt = 8;

	Tree.addBelowRoot(Tree.root, 4);
	Tree.addBelowRoot(Tree.root, 12);
	Tree.addBelowRoot(Tree.root, 2);
	Tree.addBelowRoot(Tree.root, 6);
	Tree.addBelowRoot(Tree.root, 10);
	Tree.addBelowRoot(Tree.root, 14);
	Tree.addBelowRoot(Tree.root, 1);
	Tree.addBelowRoot(Tree.root, 3);
	Tree.addBelowRoot(Tree.root, 5);
	Tree.addBelowRoot(Tree.root, 7);
	Tree.addBelowRoot(Tree.root, 9);
	Tree.addBelowRoot(Tree.root, 11);
	Tree.addBelowRoot(Tree.root, 13);
	Tree.addBelowRoot(Tree.root, 15);

	cout<< "Reverse Preorder Traversal : " ;
	bitOrder(Tree.root);
}
```

1. What is output of the program above? **(40P)**

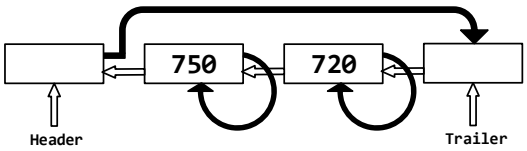
```
void addFront(const int& i)
{
   add(header->next, i);
}

void add(DoublyNode* v, int& i)
{
   DoublyNode* u    = new DoublyNode;
   u->score         = i;
   u->prev          = v->prev;
   v->prev          = u;
   v->prev->next    = u;
   u->next          = v;
}

void main()
{
   DoublyLinkedList list;
   list.addFront(750);
   list.addFront(720);
}
```

2. Which one is the list after **addFront()** function calls?
(You'll loose 5Ps from wrong answer) **(30P)**

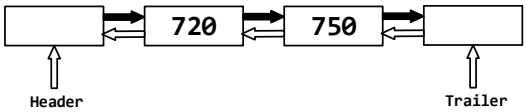
(A)



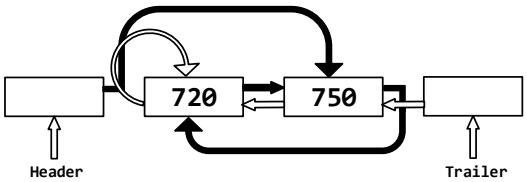
(B)



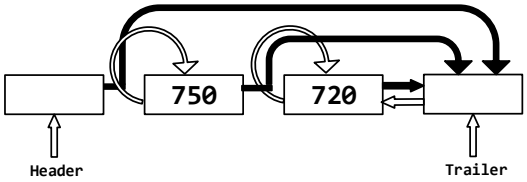
(C)

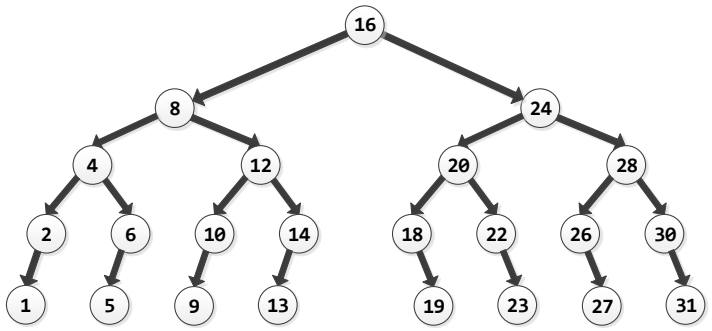


(D)



(E)






```
if (p->left != NULL)    //p points to node to be deleted
{
   temp = p->left;
   while (temp->right != NULL) temp= temp->right;
   p->elt = temp->elt;

   if (temp->left != NULL)
   {
      temp->par->right = temp->left;
      temp->left->par = temp->par;
   }
   else
   {
      temp->par->right = NULL;
   }

   delete temp;
   return;
}
```

3. Draw the tree after deleting **8** according to the code above? Draw whole tree, not only deleted part. **(30P)**