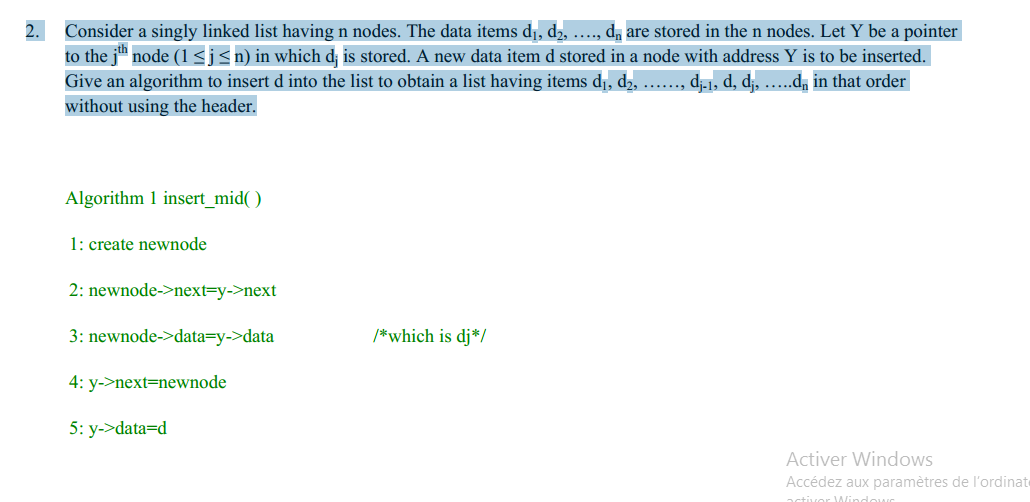
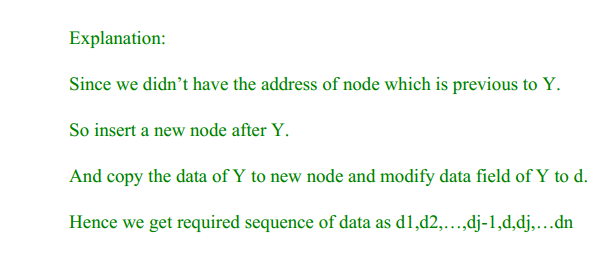
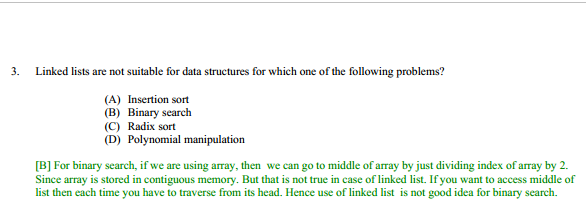
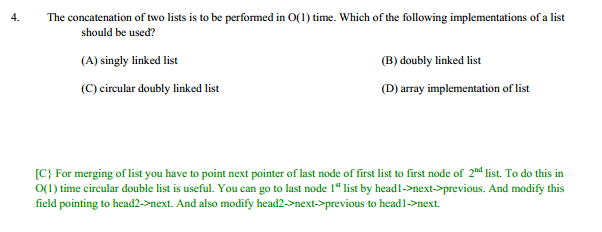


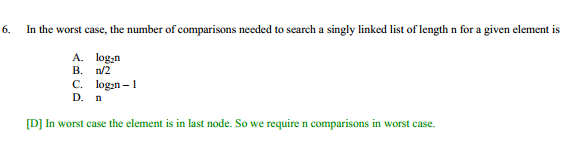
2. Envisager une liste chaînée ayant n noeuds. Les éléments de données d1, d2, ...., Dn  
sont stockés dans les n noeuds. Soit Y un pointeur vers la j ième noeud (1 ≤j ≤n) dans lequel est stocké dj. Un nouvel élément de données d stockée dans un noeud avec l'adresse Y est à insérer.  
Offrir un algorithme pour insérer d dans la liste pour obtenir une liste ayant éléments d1, d2, ......, dj-1, d, dj, ... ..dn dans cet ordre sans utiliser l'en-tête.

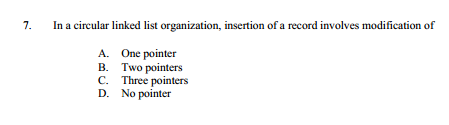


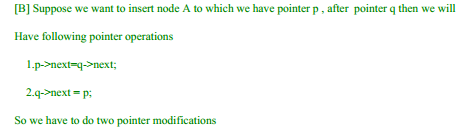


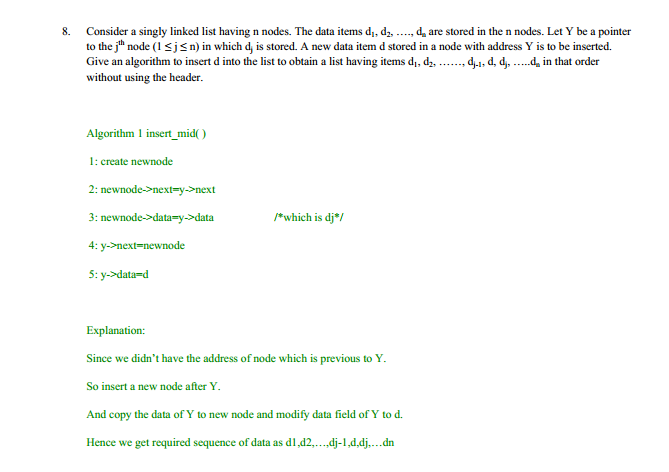


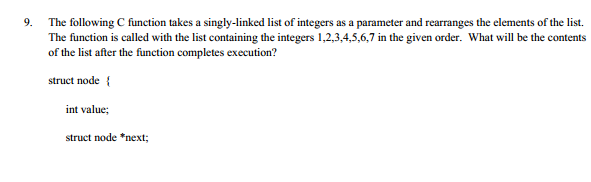


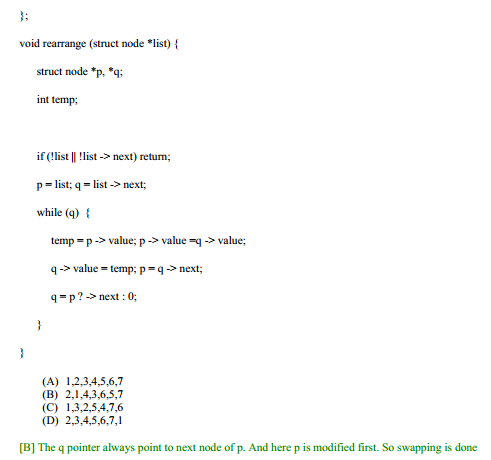












9. The following C function takes a singly-linked list of integers as a parameter and rearranges the elements of the list.

The function is called with the list containing the integers 1,2,3,4,5,6,7 in the given order. What will be the contents

of the list after the function completes execution?

struct node {

int value;

struct node \*next;

};

void rearrange (struct node \*list) {

struct node \*p, \*q;

int temp;

if (!list || !list -> next) return;

p = list; q = list -> next;

while (q) {

temp = p -> value; p -> value =q -> value;

q -> value = temp; p = q -> next;

q = p ? -> next : 0;

}

}

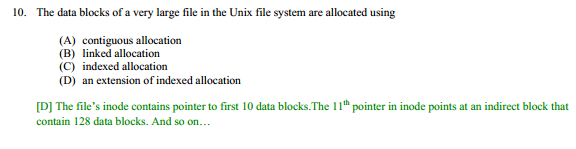
(A) 1,2,3,4,5,6,7

(B) 2,1,4,3,6,5,7

(C) 1,3,2,5,4,7,6

(D) 2,3,4,5,6,7,1

[B] The q pointer always point to next node of p. And here p is modified first. So swapping is done



11. The following C function takes a singly linked list of integers as a parameter and rearranges the elements of the list.

The list is represented as pointer to structure. The function is called with the list containing integers 1,2, 3, 4, 5, 6, 7

in the given order. What will be the contents of the list after the function completes?

struct node {int value; struct node \*next;};

void rearrange(struct node \*list) {

struct node \*p, \*q;

int temp;

if(!list || !list →next) return;

p = list; q = list →next;

while(q) {

temp = p →value;

p →value = q →value;

q →value = temp;

p = q →next;

q = p? p →next : 0;

}

}

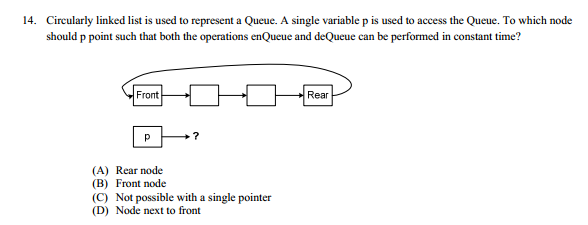
(A) 1, 2, 3, 4, 5, 6, 7 (B) 2, 1, 4, 3, 6, 5, 7

(C) 1, 3, 2, 5, 4, 7, 6 (D) 2, 3, 4, 5, 6, 7, 1

[B] The q pointer always point to next node of p. And here p is modified first. So swapping is done only

once for each

12.



[A] p points to rear node

For enQueue

1: create newnode

2: newnode->next=p->next /\*which is front node\*/

3: p->next=newnode

4: /\*rear=newnode;\*/

5: p=rear

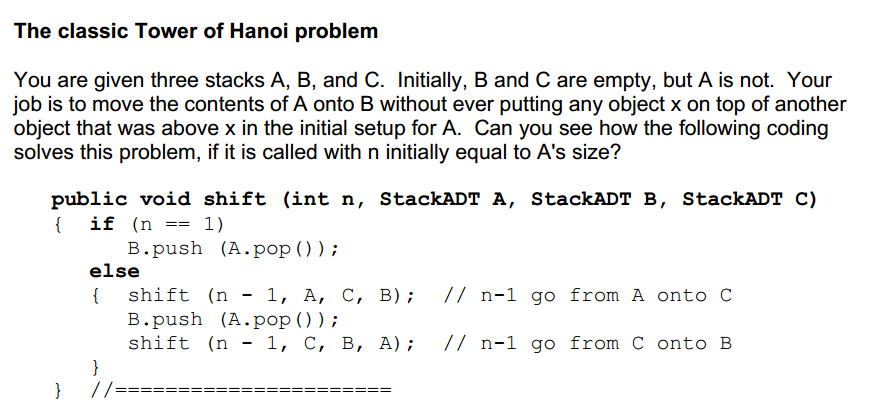
For deQueue

1:temp=p->next /\*temp is pointing to front node

2: p->next=p->next->next

3:/\* front=p->next\*/

4:delete(temp)



The classic Tower of Hanoi problem

You are given three stacks A, B, and C. Initially, B and C are empty, but A is not. Your

job is to move the contents of A onto B without ever putting any object x on top of another

object that was above x in the initial setup for A. Can you see how the following coding

solves this problem, if it is called with n initially equal to A's size?

public void shift (int n, StackADT A, StackADT B, StackADT C)

{ if(n == 1)

B.push (A.pop());

else

{ shift (n -1, A, C, B); //n-1 go from A onto C

B.push (A.pop());

shift (n -1, C, B, A); // n-1 go from C onto B

}

} //======================

Anagrams

An interesting problem is to print out all the anagrams of a given word. That is, given an

N-letter word with all letters different, print all the "words" you can form using each letter

once, regardless of whether they are actual words in some language. The number of

such rearrangements is N-factorial where N is the number of letters. The problem can be

solved in several different ways, one of which uses one queue and one stack. We

illustrate the process here and leave the coding as a major programming project.

Say the word is abcdefgh and you have just printed ehcgfdba. The very next word in

alphabetical order that you can form with those eight letters is ehdabcfg (compare the two

to see why). The way one iteration of the main loop of the process goes from a stack

containing ehcgfdbato that same stack containing ehdabcfgis as follows:

Initially you have the letters on the stack in reverse order, and you have an empty queue:

1. Repeatedly pop a value from the stack and enqueue it on the queue, stopping as soon

as you pop a value that comes alphabetically before the one you just enqueued. Call that

value the "pivot".

2. Repeatedly dequeue a value and enqueue it, stopping as soon as you dequeue a

value that comes alphabetically after the pivot. Push that value onto the stack.

3. Enqueue the pivot, then repeatedly dequeue a value and enqueue it, stopping as soon

as you dequeue a value that is alphabetically before the pivot. Push it onto the stack.

4. Repeatedly dequeue a value from the queue and push it on the stack until the queue

is empty.

Evaluating postfix expressions

Some calculators require you to enter arithmetic expressions in postfix notation. An

arithmetic postfix expressionis a sequence of numbers and operators + -\* / %

where each operator is placed directly afterthe two values it is to operate on. Some

examples of postfix expressions and the corresponding expressions in the normal infix

notationthat you are used to are as follows:

postfix notation infix notation (fully parenthesized)

7 2 - (7 -2)

3 4 + 5 \* (3 + 4) \* 5)

3 4 5 \* + (3 + (4 \* 5))

3 7 + 8 2 / - ((3 + 7) -(8 / 2))

3 4 2 6 3 / -\* + (3 + (4 \* (2 -(6 / 3))))

The reverse notation, where the operator comes directly beforethe two values it is to

operate on, is called prefix notation. It is used in the Scheme programming language.

Assume that an input string containing an arithmetic postfix expression has been read

and separated into numbers and operators stored on a queue in order. The numbers are

stored as Integer objects (with anintValue method for getting the value) and the

operators are stored as Character objects (with acharValue method for getting the

value). Then the accompanying design block is logic for evaluating the expression. The

method in the upper part of Listing 14.2 (see next page) applies this algorithm. It throws

an Exception if the data values in the queue do not form a legal postfix expression.

14.4 Implementing Stacks And Queues With Linked Lists

