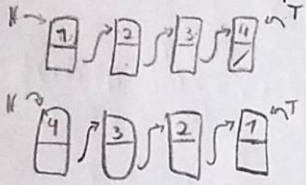
Díaz Hernández Marcos Bryan Ejercicio Virtual: 12

1: Escribe el pseudocódigo de bajo nivel para invertir los valores de una lista ligada simple, realizando una sola pasada sobre la lista.



1) Nodo tmp = Head
2) tmp = Head.next.next
3) Head.next.next = Head
4) Tail.next = tmp
5) tmp.next = Head.next.next
6) Head.next = null
7) Head = Tail
8) Tail = Head

2) Indica para los 4 tipos de listas ligadas, las principales diferencias, ventajas y desventajas que se pueden encontrar en ellas.

Las diferencias entre las listas son la forma del recorrido porque en la simple tiene Final, en la doble es posible regresar y en la circular no tiene fin pero no se puede regresar.

Las ventajas son los adecuados manejos de la memoria y el poder incrementar ó decrementar el tamaño de las listas.

Las desventajas están en el tiempo de las operaciones y de la capacidad porque al tener elementos que están enlazados es necesario tomar las consideraciones necesarios para no perder datos en los procesos o operaciones.

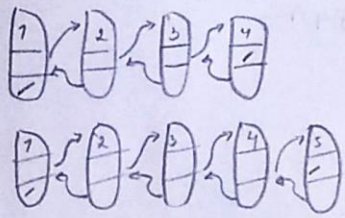
3) Explica cuáles son los dos principales problemas del almacenamiento contiguo y como se resuelven.

1) Estático: no puede cambiar de tamaño por lo que resulta poco útil al momento de añadir elementos

2) Fragmentación contigua: al momento de eliminar valores se quedan vacíos de memoria y no se pueden utilizar además de tener un tamaño específico que ocupar.

Se resuelven usando almacenamiento dinámico - ligado que sea capaz de eliminar y hacer disponibles los espacios de memoria que no se utilizan y poder hacer crecer los valores de las cadenas, usando la memoria mucho mejor.

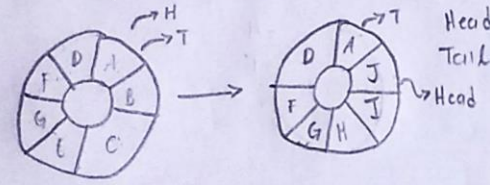
4) Escribe el pseudocódigo de bajo nivel para agregar un nodo exactamente a la mitad de una lista ligada doble de n nodos (si la lista ligada tiene una cantidad de nodos impar el nodo se deberá agregar una posición antes del que toca en medio.



1)
```
nodo tmp = head
nodo nuevo
nodo aux
if (n mod 2 == 0){
   for(i=0; i<(n/2 -1); i++){
        tmp = tmp.next;
   }
   aux = tmp.next;
   tmp.next = nuevo
   aux.prev = nuevo
   nuevo.next = aux
   nuevo.prev = tmp.
}
```

2)
```
else {
   n = n-7
   for(i=0; i<(n/2 -7); i++){
        tmp = tmp.next;
   }
   aux = tmp.next;
   tmp.next = nuevo
   aux.prev = nuevo
   nuevo.next = aux
   nuevo.prev = tmp
}
```

5) Realiza un esquema de una cola circular doble de tamaño 7 después de ejecutar las siguientes operaciones. No olvides especificar el valor de los índices.



Head = 2
Tail = 0

Si encadena el último:
H = 1
T = 0

6) Reescribe la función push de una pila y la función encolar de una cola simple de tal manera a que solo puedan ingresar elementos mayores que el anterior.

a) Push:
```
void Push(Pila *P, int x)
if (p->lista[p->tope] < x){
     p->tope = p->tope+1
     p->lista[p->tope] = x }

else
     print("No se puede")
```

b) Encolar
```
void encola(Cola *J, int x)
if(c->lista[c->ultimo] < x){
     c->ultimo = c->ultimo+1
     c->lista[c->ultimo] = x }
else
     print("No se puede")
```

7.- Explica que operaciones se tendrían que modificar en una lista ligada (de cualquier tipo)

a) Para que la lista se convierta en una pila.

La operación de añadir ya que solo se podría añadir al final y hasta un límite, la eliminación unicamente se podría al final de la lista y de un elemento a la vez los nodos se quedarían ya que ayudarían a dar estructura a la lista-pila.

b) Para que la lista se convierta en una cola simple.

La operación añadir unicamente podría añadir por el tail o al final de la lista, además al eliminar o sacar elementos solo se podría por el Head y los tendría que regresar, eso incluye el modificar los indices.

c) Para que la lista se convierta en una cola doble.

Se puede insertar y eliminar de ambos lados como la hace una lista aunque en código no haya visto la implementación debería poder seleccionar la posición y para este caso limitarse a solo insertar o eliminar al Final o principio, operaciones para indicar los indices y posiciones de los mismos.

8) Explica cómo funciona una cola de prioridad, indica cómo se definen los nodos que conforman este T.D.A.

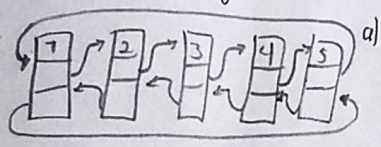
Funciona por la inserción, eliminación de elementos y la búsqueda debido a que por parámetros inserta valores en base a la asignación de valor, y coloca estos en categorías, mandando al elemento n a su categoría y al final de la misma.

```
struct Nodo{
    int dato;
    int prioridad;
    Nodo *siguiente;
};
```

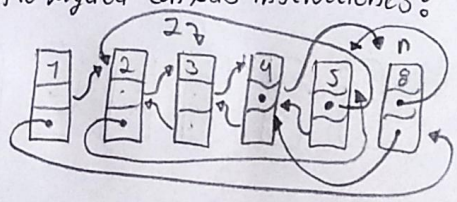
En este caso el nodo tiene los elementos normales y una forma de prioridad para asignar posición aunque revisando la aplicación por nodos, se hacía el cambio en las funciones de insertar o se usaban árboles.

9) Dada la sig. lista circular doblemente ligada con las instrucciones:

2



a) Dibuja después de realizar las operaciones.



b) Escribe de nuevo todo el código, modificando las líneas 2, 3, 9 y 10 con instrucciones diferentes que hagan lo mismo.

1) `list = list.next.next;`
2) `list.next.next.next.next.prev = list.prev.prev.prev;`
3) `list.prev.prev.prev.next = list.next.next.next.next;`
4) `Nodo n;`
5) `n.info = 8;`
6) `n.next = list.prev.prev;`
7) `n.prev = list.next;`
8) `list.prev.prev.prev.prev.next = n;`
9) `list.nat.next.next.prev = n;`
10) Con ayuda de dos pilas, evalua la siguiente expresión aritmética:

$((15 \div 3) \times (20 - 15)) + (4 \times (5 \times 8))$