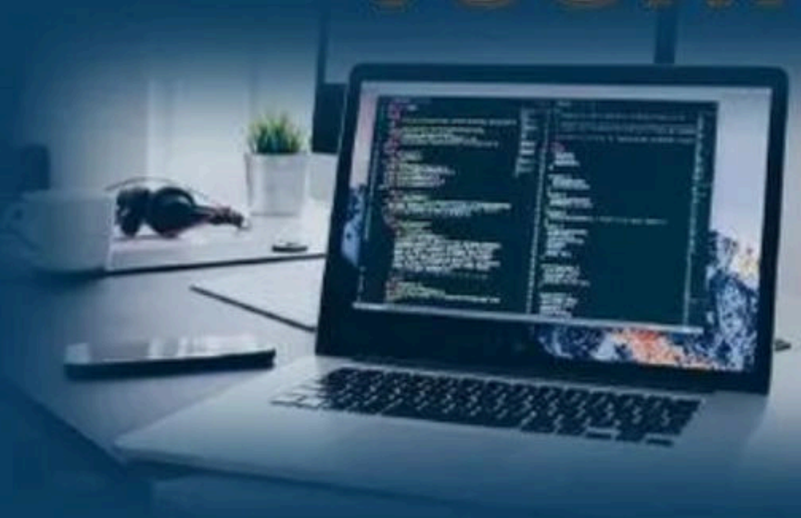
unacademy

Problem solving - Part IV

Course on Data Structure

Pankaj Sharma • Lesson 38 • Aug 26, 2024

Computer Science And Information Technology



Lecture number-38

By- Pankaj sir

unacademy

Topics

to be covered

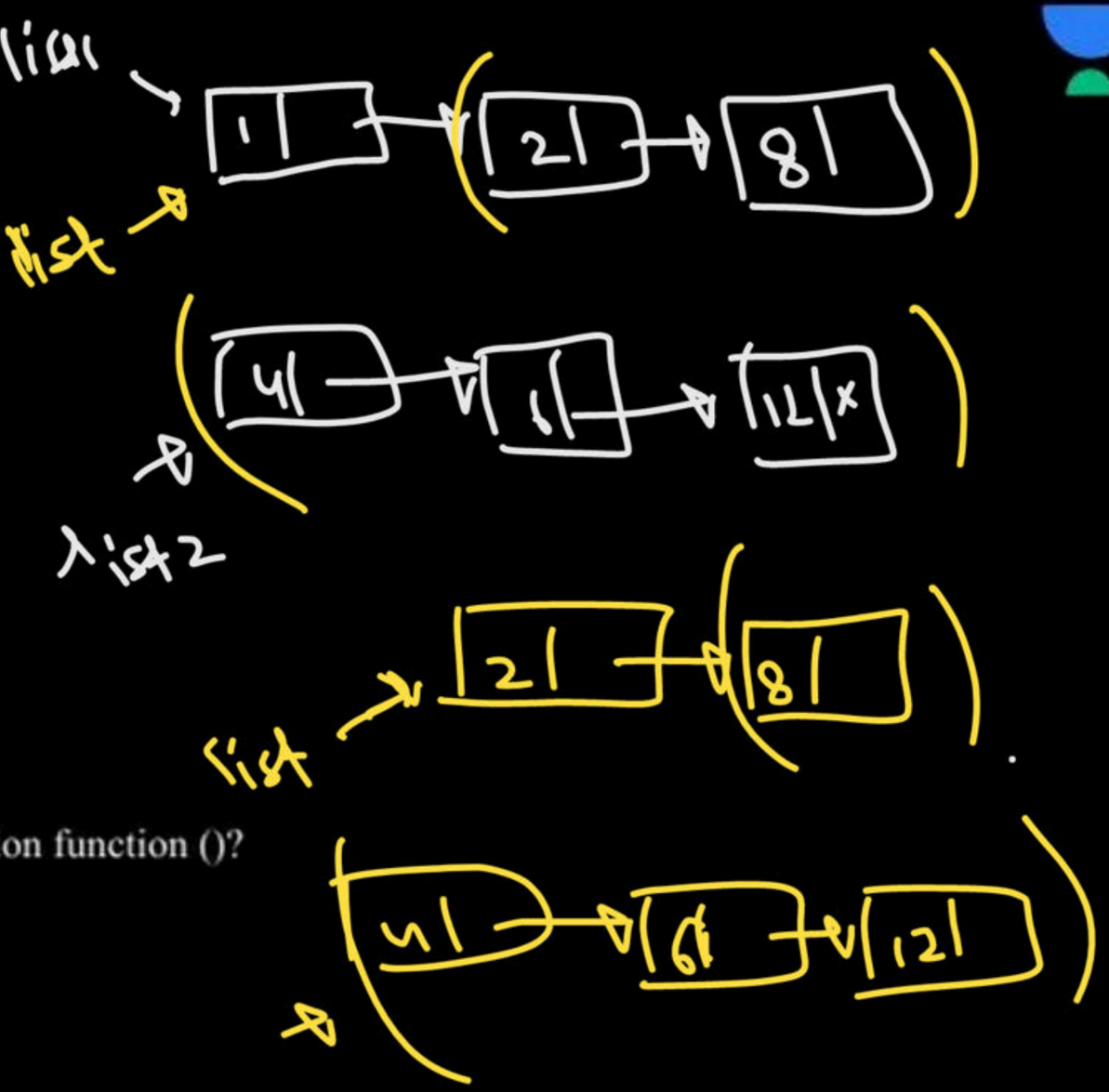
1. **Problem Solving - IV**

QUESTION

Q. Consider the following function of the two sorted linked lists:

```
struct node* function(struct node* list1, struct node* list2) {
struct node* List = NULL;
if (list1 == NULL) return (list2);
else if (list2 == NULL) return (list1);
   if (list1->data <= list2->data) {
      List = list1;
      List ->next = function (list1->next, list2);
   } else {
     List = list2;
     List ->next = function (list1, list2->next);
     }
return List;
}
```

Which of the following best describes the behavior of the function function ()?



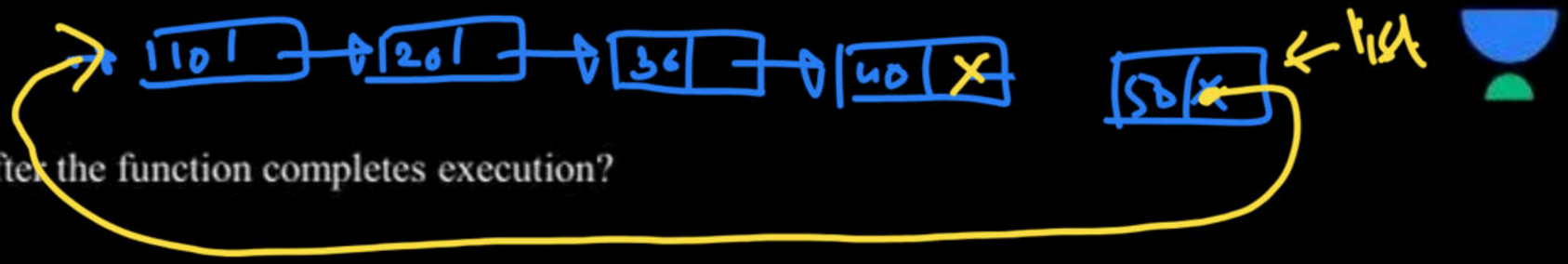
(A) Combines the two lists into a sorted list .

(B) Returns the longer of the two linked lists

(C) Returns a list that alternates nodes from the two input lists without sorting

(D) Creates a circular linked list from the two input lists

QUESTION



Q. 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) {
      if (!list || !list->next) return;
   struct node *prev = NULL, *curr = list;
   while (curr->next) {
      prev = curr;
      curr = curr->next;
   }
   curr->next = list;
   prev->next = NULL;
   list = curr;
}
```

Initial List: 10, 20, 30, 40, 50

What will be the final contents of the list?

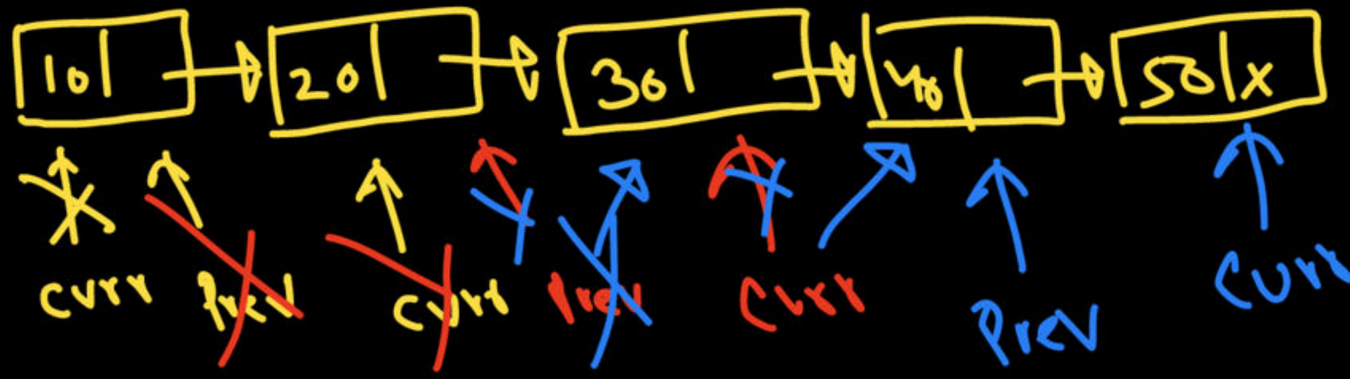
0 or 1 node

(A) 50, 10, 20, 30, 40

(B) 10, 50, 20, 30, 40

(C) 40, 50, 10, 20, 30

(D) 30, 40, 50, 10, 20



QUESTION

Q. The following C function of the two sorted singly linked lists . The function is called with two lists containing the integers in sorted order. What will be the contents of the list returned .

```
struct node {
      int value;
      struct node *next;
};
struct node* fun(struct node *list1, struct node *list2) {
   if (!list1) return list2;
   if (!list2) return list1;
   struct node *result;
   if (list1->value < list2->value) {
     result = list1;
     result->next = fun (list1->next, list2);
   } else {
      result = list2;
      result->next = fun (list1, list2->next);
   }
   return result;
}
```

List 1: 1, 3, 5

List 2: 2, 4, 6

What will be the final contents of the merged list?

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

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

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

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

1 → 2 → 3 → 4 → 5 → 6

QUESTION

Anna 24 Blanks

Q. The following C function takes a singly linked list as input and prints its elements recursively in forward order. Some part of the code is left blank.

```
typedef struct node {
    int value;
    struct node *next;
} Node;
void printlist(Node *head)
{
    if (!head) return;
    // ---- Blank ----
}
```

Choose the correct alternative to replace the blank line:

(A)
```
printlist(head->next);
printf("%d", head->data);
```

(B)
```
printf("%d", head->data);
printlist(head->next);
```

(C)
```
while (head != NULL) {
    printf("%d", head->data);
    head = head->next;
}
```

No recursion

iterative code

(D) None of these

QUESTION

Q. The following C function of a singly linked list. 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) {
  if (!list || !list->next || !list->next->next) return;
  struct node *prev = list, *curr = list->next->next;
  while (curr) {
    prev->next = curr->next;
    prev = prev->next;
    curr= prev ? prev->next : 0;
  }
}
```

Initial List: 1, 2, 3, 4, 5, 6, 7, 8, 9

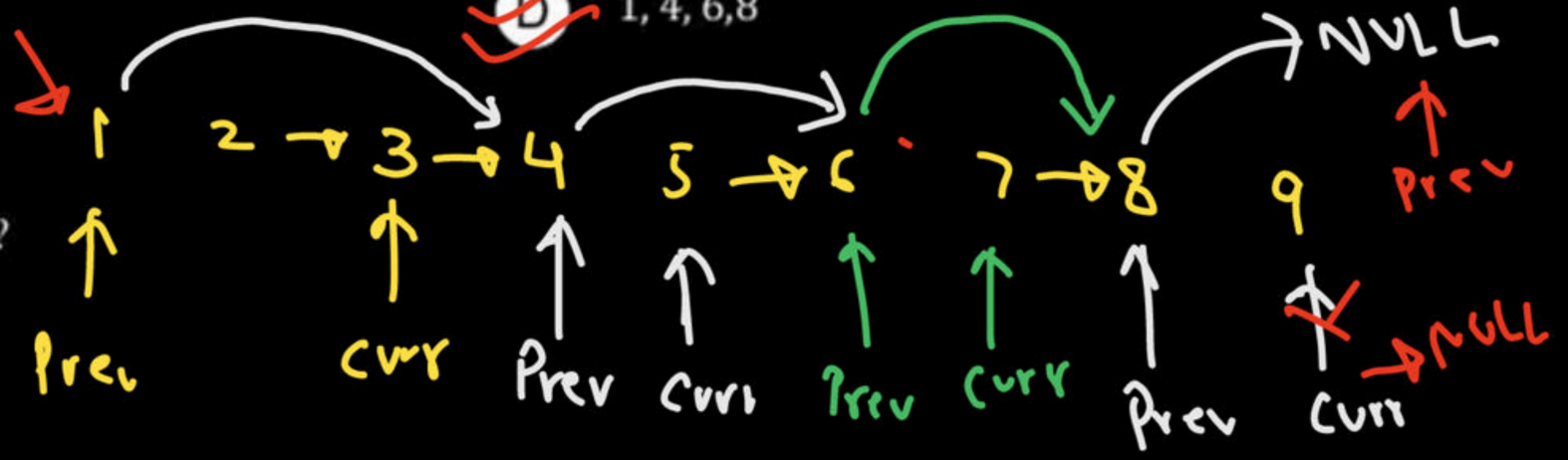
What will be the final contents of the list?

A. 1, 2, 4, 5, 7, 8

B. 1, 3, 5, 7, 9

C. 1, 2, 4, 5, 7, 8, 9

D. 1, 4, 6,8



QUESTION

Q. Consider the following statements regarding the behavior and characteristics of arrays and linked lists in different contexts:

S_1: Arrays allow random access to elements, whereas linked lists do not. True

S_2: Linked lists facilitate easier insertion and deletion of elements compared to arrays, especially when the position is known. True

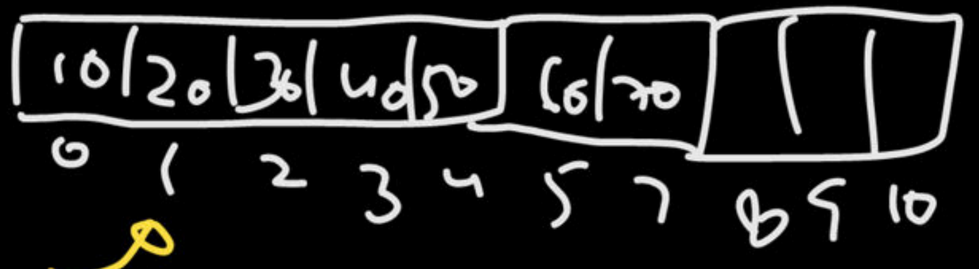
S_3: Due to better cache locality, arrays often outperform linked lists in sequential data processing. True

S_4: Insertion and deletion in a linked list always have O(1) complexity, while in arrays, these operations always require O(n) complexity. False

Which of the following combinations of statements is true?

Random Access O(1) → index, relative addressing.

Linked list: No random access.



(A) S_1, S_2, and S3 only

(B) S_1 and S_4 only

(C) S_2 and S_3 only

(D) All of the above

QUESTION

Q. Consider the following C function of a singly linked list . The function is called with a list containing the integers in the given order. What will be the contents of the list after the function completes execution?

```
struct node {
    int value;
    struct node *next;
};
struct node* fun(struct node *list, int k) {
    if (!list || k == 0) return list;
    struct node *current = list;
    int count = 1;
    while (count < k && current) {
        current = current->next;
            count++;
      }
    if (!current) return list;
    struct node *kthNode = current;
    while (current->next) current = current->next;
    current->next = list;
    list = kthNode->next;
    kthNode->next = NULL;
    return list;
}
```

Initial List: 10, 20, 30, 40, 50, 60, k = 3

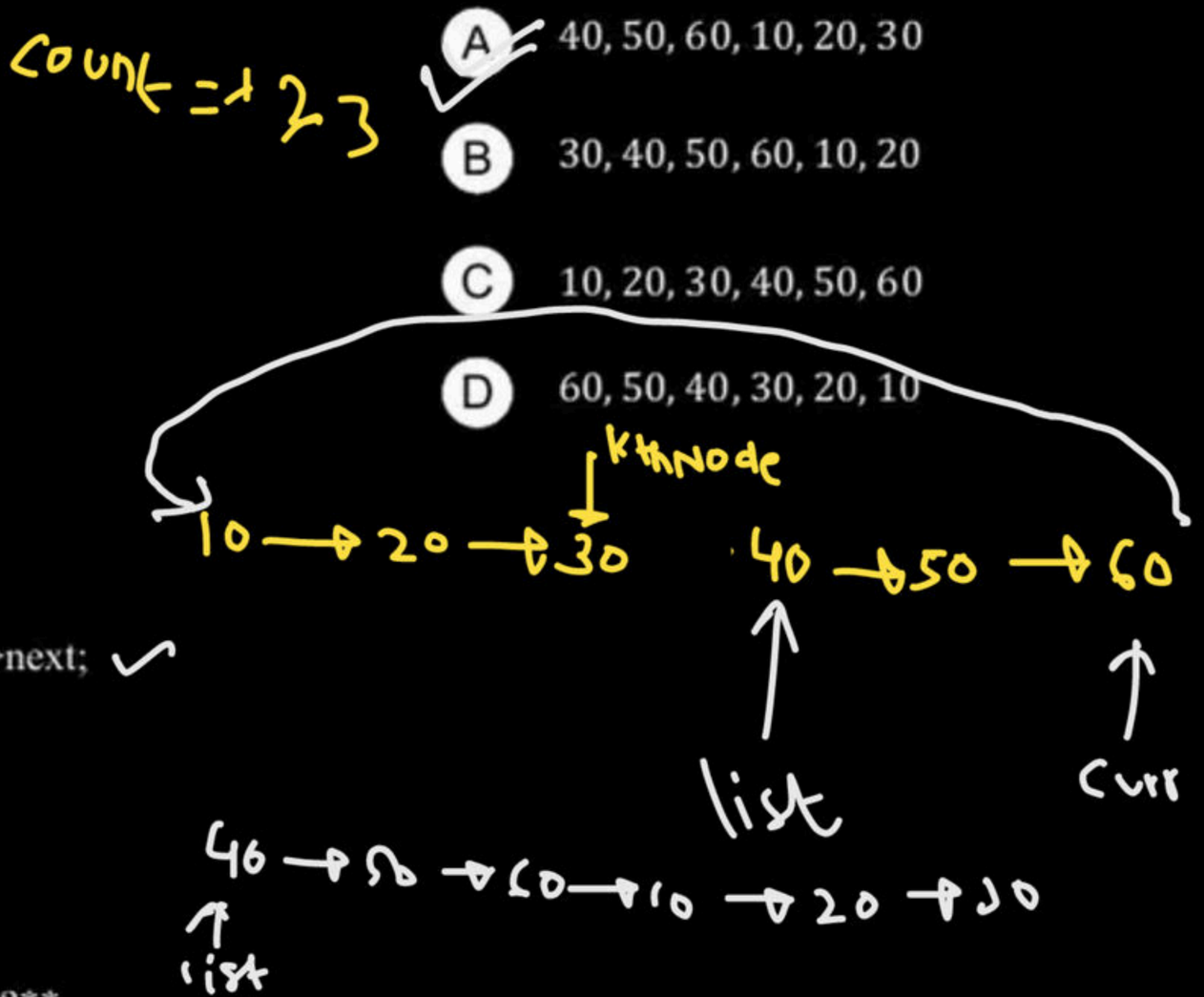
What will be the final contents of the list?

A. 40, 50, 60, 10, 20, 30

B. 30, 40, 50, 60, 10, 20

C. 10, 20, 30, 40, 50, 60

D. 60, 50, 40, 30, 20, 10



QUESTION

Q. The following C function of a singly linked list. What will be the contents of the list after the function completes execution?

```
struct node {
   int value;
   struct node *next;
};
struct node* function(struct node *head, int k)
{
   struct node *current = head;
   struct node *prev = NULL, *next = NULL;
   int count = 0;
   while (current && count < k)
   {
      next = current->next;
      current->next = prev;
      prev = current;
      current = next;
      count++;
   }
   if (next) head->next = function(next, k);
   return prev;
}
```

Initial List: 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, k = 4
What will be the final contents of the list?

(A) 4, 3, 2, 1, 8, 7, 6, 5, 10, 9

(B) 3, 2, 1, 4, 7, 6, 5, 8, 9, 10

(C) 4, 3, 2, 1, 5, 6, 7, 8, 9, 10

(D) 1, 2, 3, 4, 5, 6, 7, 8, 9, 10

QUESTION

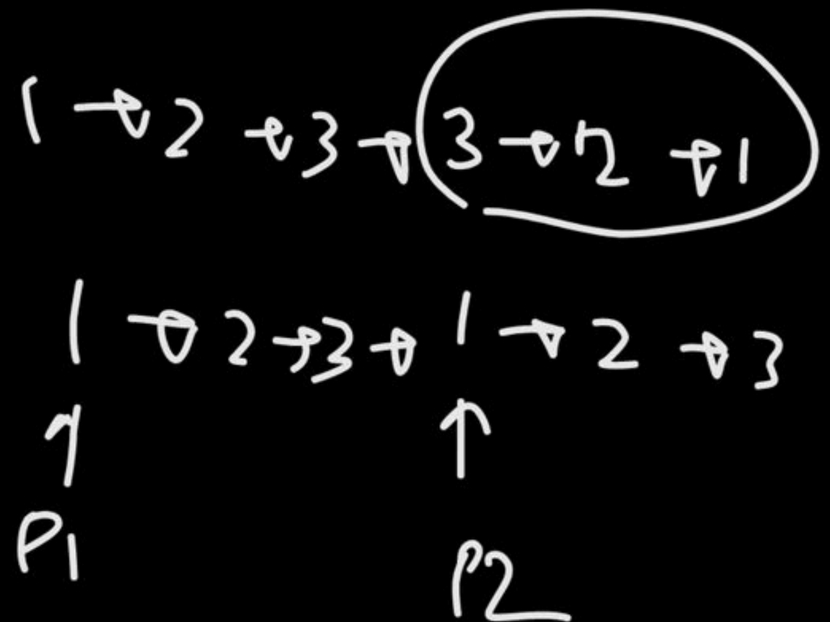
Q. For a singly linked list P, what is the time complexity of the best algorithm to determine whether the list is a palindrome?

- A $O(n^2)$
- B $O(n \log n)$
- C $O(n)$
- D $O(\log n)$

Reverse → O(n)

Middle → O(n)

a) Find the middle
b) Reverse the second half
c) Compare the two halfes
d) Restore the original list (optional)



QUESTION

Q. Consider the following C function of a sorted singly linked list. What will be the contents of the list after the function completes execution?

```
struct node {
   int value;
   struct node *next;
};
void random(struct node *list) {
   struct node *current = list;
   while (current && current->next) {
      if (current->value == current->next->value) {
         struct node *temp = current->next;
         current->next = current->next->next;
         free(temp);
      } else {
         current = current->next;
      }
   }
}
```

Initial List: 1, 1, 2, 3, 3, 4, 5, 5

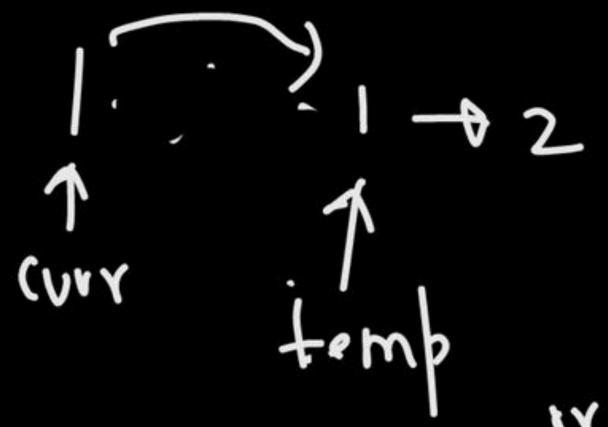
What will be the final contents of the list?

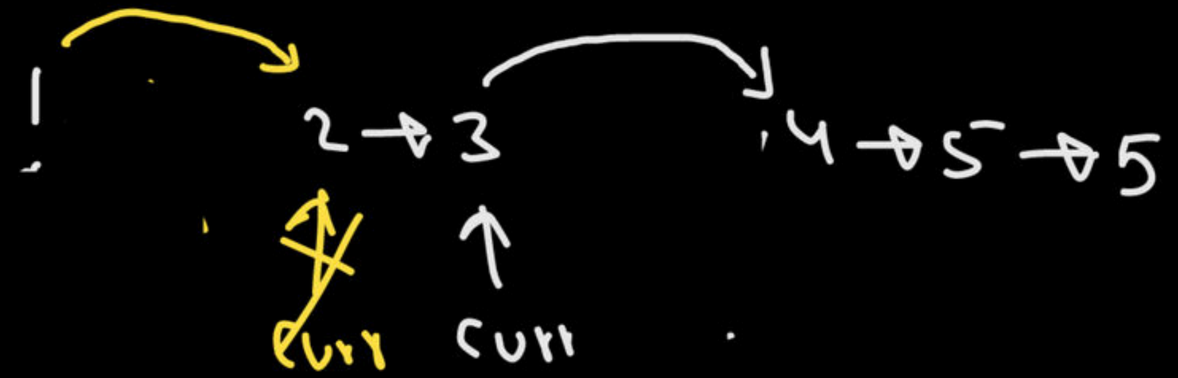
(A) 1, 2, 3, 4, 5

(B) 2, 3, 4, 5

(C) 1, 3, 4, 5

(D) 1, 2, 4, 5





QUESTION

Q. Consider the following C function designed to merge two sorted linked lists:

```
struct node* DoSomething(struct node* first, struct node* second) {
    struct node* temp= NULL;
    if (first == NULL) return second;
    else if (second == NULL) return first;
    if (first->data < second->data) {
        temp = first;
        temp ->next = DoSomething (first->next, second);
    } else {
        temp = second;
        temp ->next = DoSomething (first, second->next);
    }
    return temp;
}
```

Which of the following operations does the function DoSomething () perform?

A. Merges two sorted linked lists while maintaining their sorted order

B. Merges two sorted linked lists but reverses the order of elements

C. Finds and returns the common elements in both lists

D. Creates a doubly linked list from the two input lists

QUESTION

Q. Consider The following C function of the singly linked list. What will be the contents of the list after the function completes execution?

```
struct node {
    int value;
    struct node *next;
};
struct node* fun(struct node *list) {
    struct node *prev = NULL, *current = list, *next = NULL;
    while (current != NULL) {
        next = current->next;
        current->next = prev;
        prev = current;
        current = next;
    }
    return prev;
}
```

Initial List: 10, 20, 30, 40, 50

What will be the final contents of the list?

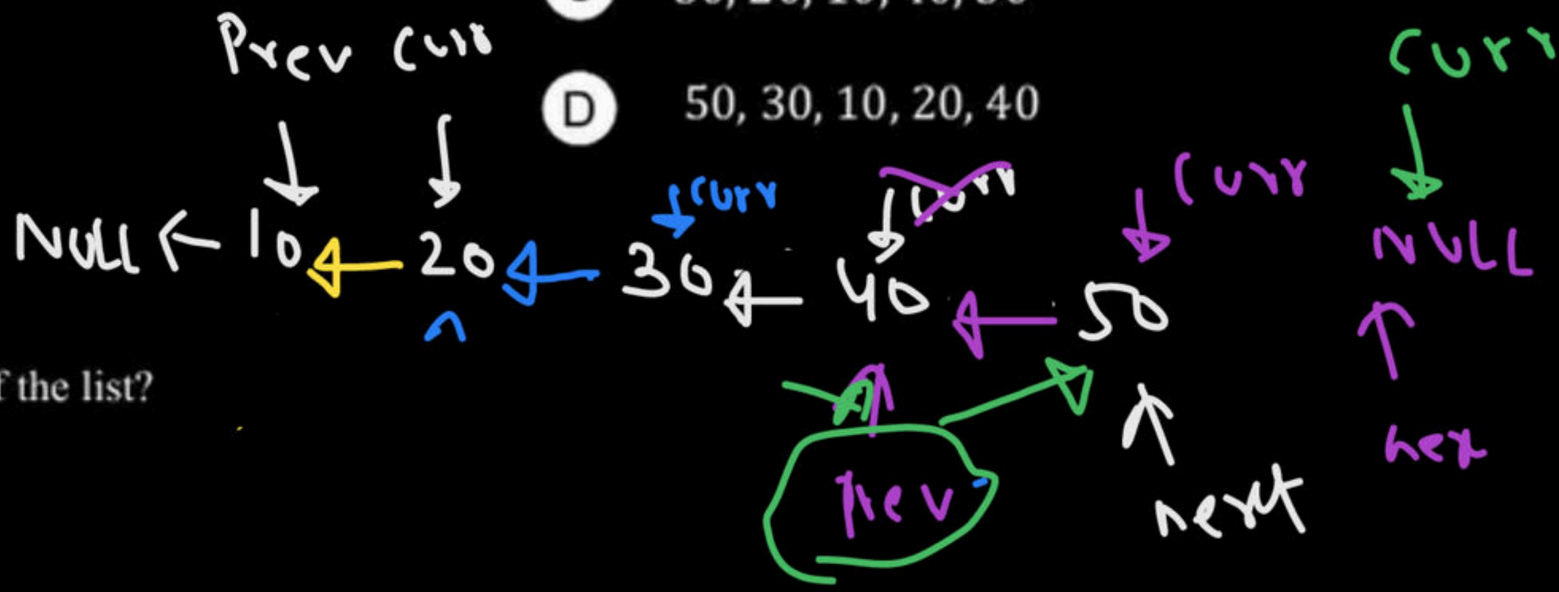
(A) 50, 40, 30, 20, 10

(B) 10, 20, 30, 40, 50

(C) 30, 20, 10, 40, 50

(D) 50, 30, 10, 20, 40

Iterative code to reverse a given LL



QUESTION

Q. Consider The following C function of a singly linked list. What will be the contents of the list after the function completes execution?

```
struct node {
    int value;
    struct node *next;
};
struct node* func(struct node *list)
{
    if (!list || !list->next) return NULL;
    struct node *slow = list, *fast = list, *prev = NULL;
    while (fast && fast->next)
        {
            fast = fast->next->next;
            prev = slow;
            slow = slow->next;
        }
      prev->next = slow->next;
      free(slow);
      return list;
}
```

Initial List: 1, 2, 3, 4, 5, 6

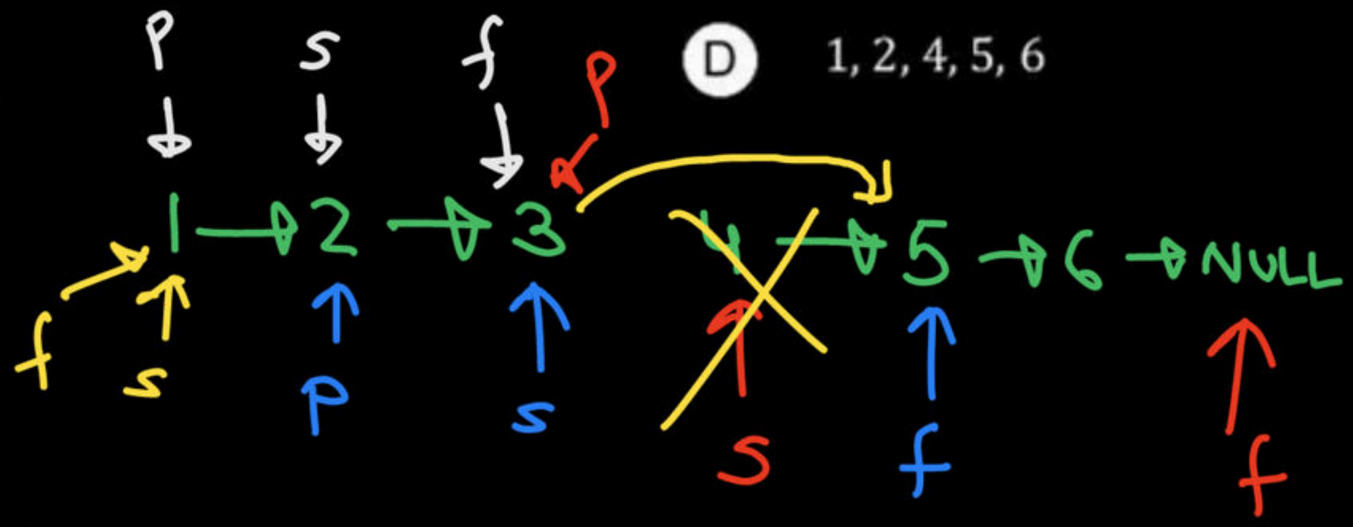
What will be the final contents of the list?

A. 1, 2, 3, 5, 6

B. 2, 3, 4, 5, 6

C. 1, 3, 4, 5, 6

D. 1, 2, 4, 5, 6



QUESTION

Q. Evaluate the following statements about arrays and linked lists in terms of their structural and operational efficiencies:

S_1: Linked lists cannot provide O(1) time complexity for accessing elements, while arrays can, due to their continuous memory allocation.

S_2: Arrays are preferable when dealing with operations requiring frequent random access, whereas linked lists are more suitable when frequent insertions and deletions are necessary.

S_3: Cache performance is generally superior in arrays compared to linked lists due to the contiguous memory layout of arrays.

S_4: Insertion at the beginning of an array requires O(1) time complexity, while it takes O(n) in a linked list.

Which of the following are correct?

(A) S_1, S_2, and S_3 only

(B) S_1 and S_4 only

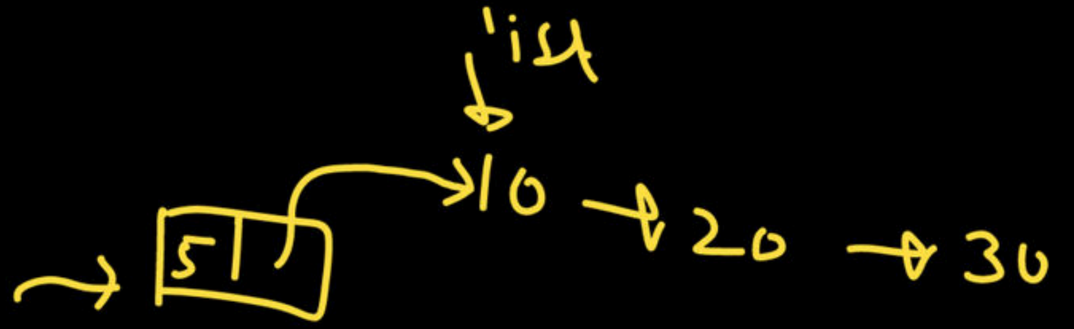
(C) S_2 and S_3 only

(D) All of the above

QUESTION

Q. Consider The following C function of a singly linked list. What will be the contents of the list after the function completes execution?

```
struct node {
    int value;
    struct node *next;
};
struct node* fun(struct node *list, int newValue)
{
    struct node *newNode = (struct node*) malloc(sizeof(struct node));
    newNode->value = newValue;
    newNode->next = list;
    return newNode;
}
```



Initial List: 10, 20, 30
New Value: 5
What will be the final contents of the list?

A. 5, 10, 20, 30

B. 10, 20, 30, 5

C. 10, 5, 20, 30

D. 5, 30, 20, 10

QUESTION

Q.

What will be the contents of the merged list?

```
struct node {
      int value;
      struct node *next;
};
struct node* DoSomething(struct node *list1, struct node *list2) {
   struct node dummy;
   struct node *tail = &dummy;
   dummy.next = NULL;
   while (list1 && list2) {
      if (list1->value <= list2->value) {
         tail->next = list1;
         list1 = list1->next;
      } else {
         tail->next = list2;
         list2 = list2->next;
      }
      tail = tail->next;
   }
  tail->next = list1 ? list1 : list2;
return dummy.next;
}
```

List 1: 1, 3, 5

List 2: 2, 4, 6

A 1, 2, 3, 4, 5, 6

B 6, 5, 4, 3, 2, 1

C 1, 3, 2, 4, 5, 6

D 1, 4, 2, 3, 5, 6

unacademy

unacademy

Review

THANK YOU!

Here's to a cracking journey ahead!