Rajalakshmi Engineering College

Name: KEERTHANA S
Email: 240801161@rajalakshmi.edu.in
Roll no: 240801161
Phone: 9345818052
Branch: REC
Department: I ECE FB
Batch: 2028
Degree: B.E - ECE

Scan to verify results

NeoColab_REC_CS23231_DATA STRUCTURES

REC_DS using C_Week 2_MCQ_Updated

Attempt : 2
Total Mark : 20
Marks Obtained : 20

Section 1 : MCQ

1. Which pointer helps in traversing a doubly linked list in reverse order?

Answer

prev

Status : Correct *Marks : 1/1*

2. What does the following code snippet do?

```
struct Node* newNode = (struct Node*)malloc(sizeof(struct Node));
newNode->data = value;
newNode->next = NULL;
newNode->prev = NULL;
```

Answer

Creates a new node and initializes its data to 'value'

Status : Correct ***Marks : 1/1***

3. Which of the following statements correctly creates a new node for a doubly linked list?

Answer

struct Node* newNode = (struct Node*) malloc(sizeof(struct Node));

Status : Correct ***Marks : 1/1***

4. Where Fwd and Bwd represent forward and backward links to the adjacent elements of the list. Which of the following segments of code deletes the node pointed to by X from the doubly linked list, if it is assumed that X points to neither the first nor the last node of the list?

A doubly linked list is declared as

```
struct Node {
      int Value;
      struct Node *Fwd;
      struct Node *Bwd;
);
```

Answer

X-&gt;Bwd-&gt;Fwd = X-&gt;Fwd; X-&gt;Fwd-&gt;Bwd = X-&gt;Bwd;

Status : Correct ***Marks : 1/1***

5. Which code snippet correctly deletes a node with a given value from a doubly linked list?

```
void deleteNode(Node** head_ref, Node* del_node) {
   if (*head_ref == NULL || del_node == NULL) {
      return;
   }
   if (*head_ref == del_node) {
      *head_ref = del_node->next;
```

```
    }
    if (del_node->next != NULL) {
        del_node->next->prev = del_node->prev;
    }
    if (del_node->prev != NULL) {
        del_node->prev->next = del_node->next;
    }
    free(del_node);
}
```

Answer

Deletes the first occurrence of a given data value in a doubly linked list.

Status : Correct ***Marks : 1/1***

6. Which of the following is false about a doubly linked list?

Answer

Implementing a doubly linked list is easier than singly linked list

Status : Correct ***Marks : 1/1***

7. What will be the output of the following code?

```
#include <stdio.h>
#include <stdlib.h>

struct Node {
    int data;
    struct Node* next;
    struct Node* prev;
};

int main() {
    struct Node* head = NULL;
    struct Node* temp = (struct Node*)malloc(sizeof(struct Node));
    temp->data = 2;
    temp->next = NULL;
    temp->prev = NULL;
    head = temp;
    printf("%d\n", head->data);
    free(temp);
    return 0;
}
```

Answer

2

Status : Correct ***Marks : 1/1***

8. How do you reverse a doubly linked list?

Answer

By swapping the next and previous pointers of each node

Status : Correct ***Marks : 1/1***

9. How many pointers does a node in a doubly linked list have?

Answer

2

Status : Correct ***Marks : 1/1***

10. What is the correct way to add a node at the beginning of a doubly linked list?

Answer

void addFirst(int data){   Node* newNode = new Node(data);   newNode-&gt;next = head;   if (head != NULL) {   head-&gt;prev = newNode;   }   head = newNode;   }

Status : Correct ***Marks : 1/1***

11. Consider the following function that refers to the head of a Doubly

Linked List as the parameter. Assume that a node of a doubly linked list has the previous pointer as prev and the next pointer as next.

Assume that the reference of the head of the following doubly linked list is passed to the below function 1 <--> 2 <--> 3 <--> 4 <--> 5 <-->6. What should be the modified linked list after the function call?

```
Procedure fun(head_ref: Pointer to Pointer of node)
  temp = NULL
  current = *head_ref

  While current is not NULL
    temp = current->prev
    current->prev = current->next
    current->next = temp
    current = current->prev
  End While

  If temp is not NULL
    *head_ref = temp->prev
  End If
End Procedure
```

Answer

6 &lt;--&gt; 5 &lt;--&gt; 4 &lt;--&gt; 3 &lt;--&gt; 2 &lt;--&gt; 1.

Status : Correct ***Marks : 1/1***

12. What is a memory-efficient double-linked list?

Answer

A doubly linked list that uses bitwise AND operator for storing addresses

Status : Correct ***Marks : 1/1***

13. Consider the provided pseudo code. How can you initialize an empty two-way linked list?

```
Define Structure Node
   data: Integer
   prev: Pointer to Node
   next: Pointer to Node
End Define

Define Structure TwoWayLinkedList
   head: Pointer to Node
   tail: Pointer to Node
End Define
```

Answer

struct TwoWayLinkedList* list = malloc(sizeof(struct TwoWayLinkedList)); list-&gt;head = NULL; list-&gt;tail = NULL;

Status : Correct ***Marks : 1/1***

14. How do you delete a node from the middle of a doubly linked list?

Answer

All of the mentioned options

Status : Correct ***Marks : 1/1***

15. What will be the effect of setting the prev pointer of a node to NULL in a doubly linked list?

Answer

The node will become the new head

Status : Correct ***Marks : 1/1***

16. What is the main advantage of a two-way linked list over a one-way linked list?

Answer

Two-way linked lists allow for traversal in both directions.

Status : Correct ***Marks : 1/1***

17. What happens if we insert a node at the beginning of a doubly linked list?

Answer

The previous pointer of the new node is NULL

Status : Correct ***Marks : 1/1***

18. Which of the following is true about the last node in a doubly linked list?

Answer

Its next pointer is NULL

Status : Correct ***Marks : 1/1***

19. Which of the following information is stored in a doubly-linked list’s nodes?

Answer

All of the mentioned options

Status : Correct ***Marks : 1/1***

20. What will be the output of the following program?

```
#include <stdio.h>
#include <stdlib.h>

struct Node {
   int data;
   struct Node* next;
   struct Node* prev;
};
```

```
int main() {
    struct Node* head = NULL;
    struct Node* tail = NULL;
    for (int i = 0; i < 5; i++) {
        struct Node* temp = (struct Node*)malloc(sizeof(struct Node));
        temp->data = i + 1;
        temp->prev = tail;
        temp->next = NULL;
        if (tail != NULL) {
            tail->next = temp;
        } else {
            head = temp;
        }
        tail = temp;
    }
    struct Node* current = head;
    while (current != NULL) {
        printf("%d ", current->data);
        current = current->next;
    }
    return 0;
}
```

Answer

1 2 3 4 5

Status : Correct ***Marks : 1/1***