Jadavpur University

Department of Computer Science & Engg

MCA, 1st Year, 2nd Semester, 2021

Data and File Structures, Class Test

Time: 1 hour **Maximum Marks: 30**

All questions are compulsory. Each question has five choices having only one option correct. You have to justify the correctness of the choice you have selected. Marks distribution: for each question, 0.5 for the correct choice, and 1.5 for explaining such choice. If you find none of the options is correct, you should write the correct answer to get full credit for that question.

---

1. The recurrence relation capturing the optimal time of the Tower of Hanoi problem with n discs is

- **a.** $T(n) = 2T(n – 2) + 2$
- **b.** $T(n) = 2T(n – 1) + n$
- **c.** $T(n) = 2T(n – 1) + 1$
- **d.** $T(n) = 2T(n/2) + 1$
- **e.** None of the above

2. What does the following function do for a given Linked List with first node as head?

```
void fun1(struct node* head)
{
if(head->next == NULL)
return;
fun1(head->next);
printf("%d ", head->data);
}
```

- **a.** Prints all nodes of linked lists
- **b.** Prints all nodes of linked list in reverse order
- **c.** Prints alternate nodes of Linked List
- **d.** Prints alternate nodes in reverse order
- **e.** None of the above

3. Consider the following function

```
void fun(struct node **head_ref)
{
struct node *temp = NULL;
struct node *current = *head_ref;
while (current != NULL)
{
temp = current->prev;
current->prev = current->next;
current->next = temp;
current = current->prev;
}
if(temp != NULL )
*head_ref = temp->prev;
}
```

Assume that reference of head of following doubly linked list is passed to above function

1 <--> 2 <--> 3 <--> 4 <--> 5 <-->6. What should be the output

a. 2 <--> 1 <--> 4 <--> 3 <--> 6 <-->5
b. 5 <--> 4 <--> 3 <--> 2 <--> 1 <-->6
c. 6 <--> 5 <--> 4 <--> 3 <--> 1 <--> 2
d. 6 <--> 5 <--> 4 <--> 3 <--> 2 <--> 1
e. None of the above

4. The following function reverse() is supposed to reverse a singly linked list. There is one line written by a programmer at the end of the function.

```
/* Link list node */
struct node
{
    int data;
    struct node* next;
};
/* head_ref is a double pointer that points to the head (or start) pointer
  of linked list */
static void reverse(struct node** head_ref)
{
    struct node* prev   = NULL;
    struct node* current = *head_ref;
    struct node* next;
    while (current != NULL)
    {
        next  = current->next;
        current->next = prev;
        prev = current;
        current = next;
    }
    /*ADD A STATEMENT HERE*/
    *head_ref = prev;
}
```

What should be added in place of "/*ADD A STATEMENT HERE*/" so that the function correctly reverses a linked list.

a. *head_ref = prev;
b. *head_ref = current;
c. *head_ref = next;
d. *head_ref = NULL;
e. None of the above

5. What is the output of following function for start pointing to first node of following linked list? 1->2->3->4->5->6

```
void fun(struct node* start)
{
  if(start == NULL)
    return;
  printf("%d  ", start->data);

  if(start->next != NULL )
    fun(start->next);
```

```
  printf("%d  ", start->data);
}
```

a. 1 4 6 6 4 1
b. 1 3 5 1 3 5
c. 1 2 3 4 4 3 2 1
d. 1 3 5 5 3 1
e. None of the above

6. If

$f(n) = 3n^{\sqrt{n}}$

$g(n) = 2^{\sqrt{n}\log_2 n}$

$h(n) = n!$

a. h(n) is O(f(n))
b. h(n) is O(g(n))
c. g(n) is not O(f(n))
d. f(n) is O(g(n))
e. None of the above

7. The following C function takes a simply-linked list as an input argument. It modifies the list by moving the last element to the front of the list and returns the modified list. Some part of the code is left blank. Choose the correct alternative to replace the blank line.

```
typedef struct node
{
  int value;
  struct node *next;
}Node;

Node *move_to_front(Node *head)
{
  Node *p, *q;
  if ((head == NULL: || (head->next == NULL))
    return head;
  p = NULL; q = head;
  while (p-> next !=NULL)
  {
    q = p;
    p = p->next;
  }
  _____________________________
  return head;
}
```

a. q = NULL; p->next = head; head = p;
b. q->next = NULL; head = p; p->next = head;
c. head = p; p->next = q; q->next = NULL;
d. q->next = NULL; p->next = head; head = p;
e. None of the above

8. The following C function takes a single-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?p->next:0;
  }
}
```

a. 1,2,3,4,5,6,7
b. 2,1,3,4,6,5,7
c. 1,3,2,5,4,7,6
d. 2,1,4,3,6,5,7
e. None of the above

9. In the worst case, the number of comparisons needed to search a singly linked list of length n for a given element is

a. lg n
b. n/2
c. lg (n – 1)
d. n+1
e. None of the above

10. The sorting method, which sorts a set of elements that are already in order or in reverse order with equal speed, is

a. Quicksort
b. Heap sort
c. Bubble sort
d. Merge sort
e. None of the above

11. Consider the function f defined below.

```
struct item
{
int data;
struct item * next;
};

int f(struct item *p)
{
return (
(p == NULL) ||
(p->next == NULL) ||
(( P->data->next >= p->data) && f(p->next))
);
}
```

For a given linked list p, the function f returns 1 if and only if

- **a.** the list is empty or has exactly one element
- **b.** the elements in the list are sorted in non-decreasing order of data value
- **c.** the elements in the list are sorted in non-increasing order of data value
- **d.** not all elements in the list have the same data value
- **e.** None of the above

12. What are the time complexities of finding the 10th element from the end and the 10th element from the beginning in a singly linked list? Let n be the number of nodes in a linked list; you may assume n > 8.

- **a.** O(1) and O(n)
- **b.** O(10) and O(10)
- **c.** O(n) and O(1)
- **d.** O(n) and O(n)
- **e.** None of the above

13. Consider the following function to traverse a linked list.

```
void traverse(struct Node *head)
{
while (head->next != NULL)
{
printf("%d ", head->data);
head = head->next;
}
}
```

Which of the following is FALSE about the above function?

- **a.** The function may crash when the linked list is not empty
- **b.** The function prints elements present in the linked list
- **c.** The function is implemented incorrectly because it changes head
- **d.** All of the above
- **e.** None of the above

14. Following is C like pseudo-code of a function that takes a number as an argument and uses a stack S to do the processing.

```
void fun(int n)
{
Stack S; // Say it creates an empty stack S
while (n > 0)
{
// This line pushes the value of n%2 to stack S
push(&S, n/2);

n = n%2;
}

// Run while Stack S is not empty
while (!isEmpty(&S))
printf("%d ", pop(&S)); // pop an element from S and print it
}
```

What does the above function do in general?

- **a.** Prints binary representation of n in reverse order
- **b.** Prints binary representation of n
- **c.** Prints the value of Logn
- **d.** Prints the value of Logn in reverse order
- **e.** None of the above

15. Let x, y, and z denote the last node traversed in pos, in, and pre-order access, respectively, of a binary tree. Which of the following is always true?

- **a.** y = x
- **b.** y= z
- **c.** z=x
- **d.** All of the above
- **e.** None of the above