Questions for Question Bank

21CSC201J: Data Structures and Algorithms

Unit – II: Implementation of Circular Linked List

Prepared by: Dr. R.Madhura

Q1 – Q5: Objective Type Questions

1. **In a circular linked list**

   a) Components are all linked together in some sequential manner.
   b) There is no beginning and no end.
   c) Components are arranged hierarchically.
   d) Forward and backward traversal within the list is permitted.
   Answer: b

2. **A variant of the linked list in which none of the node contains NULL pointer is?**

   a) Singly linked list
   b) Doubly linked list
   c) Circular linked list
   d) None

   Answer: c

3. **In circular linked list, insertion of node requires modification of?**

   a) One pointer
   b) Two pointer
   c) Three pointer
   d) None

   Answer: b

4. A **circular linked list can be used for**

   a) Stack
   b) Queue
   c) both stack & queue
   d) neither stack or queue

   Answer: c

5. **Which of the following application makes use of a circular linked list?**

   a) Undo operation in a text editor
   b) Recursive function calls
   c) Allocating CPU to resources
   d) All of the mentioned

   Answer: c

Understanding question

6. **In a circular linked list, is it possible to find out if the next element of a pointer points to itself or not? If yes, then how?**

   Yes, it is possible to find out if the next element of a pointer points to itself or not. To do this, you can simply check if the pointer's next element is equal to the pointer itself. If it is, then the pointer is pointing to itself and the linked list is circular.

7. **Write a program to delete the front node in the circular linked list**

   We conventionally delete the front node from the list in this program. To delete a node, we need to check if the list is empty. If it is not empty then point the rear node to the front->next and rear->next to front. This removes the first node.

```
void del()
{
temp=front;
if(front==NULL)
printf("\nUnderflow :");
else
{
if(front==rear)
{
printf("\n%d",front->info);
front=rear=NULL;
}
else
{
printf("\n%d",front->info);
front=front->next;
```

```
rear->next=front;
}

temp->next=NULL;
free(temp);
}
}
```

8. Convert a singly linked list in to circular linked list

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

typedef struct list {
   int data;
   struct list* next;
} node;

void display(node* temp)
{
   //now temp1 is head basically
   node* temp1 = temp;
   printf("The list is as follows :\n%d->", temp->data);
   temp = temp->next;
   //which not circle back to head node
   while (temp != temp1) {
      printf("%d->", temp->data);
      temp = temp->next;
   }
   printf("%d\n", temp1->data);

   return;
}

// Main Code
int main()
{
   node *head = NULL, *temp, *temp1;
   int choice, count = 0, key;

   //Taking the linked list as input
   do {
      temp = (node*)malloc(sizeof(node));
      if (temp != NULL) {
         printf("\nEnter the element in the list : ");
```

```
            scanf("%d", &temp->data);
            temp->next = NULL;
            if (head == NULL) {
               head = temp;
            }
            else {
               temp1 = head;
               while (temp1->next != NULL) {
                  temp1 = temp1->next;
               }
               temp1->next = temp;
            }
         }
         else {
            printf("\nMemory not avilable...node allocation is not possible");
         }
         printf("\nIf you wish to add m ore data on the list enter 1 : ");
         scanf("%d", &choice);
      } while (choice == 1);

      //In order to convert a singly linked list to a
      //circular singly linked list we just need to copy
      //the address of the head node to the next of the last node
      temp = head;

      //travarsing to the last node
      while (temp->next != NULL) {
         temp = temp->next;
      }

      //the address of the head is copied to the next part
      //of the last node.....now it is a circular singly linked list
      temp->next = head;

      display(head);

      return 0;
   }
```

9. Are there any advantages or disadvantages associated with circular doubly-linked lists over regular doubly-linked lists?

One advantage of a circular doubly-linked list is that it can be easier to implement certain algorithms with them. For example, it is easier to traverse a circular doubly-linked list in both directions than it is a regular doubly-linked

list. However, one disadvantage of a circular doubly-linked list is that it can be more difficult to keep track of the head and tail nodes, since there is no beginning or end to the list.

Scenario Based

10. **The concatenation of two lists is to be performed in O(1) time. Which list should be used?**

Circular Doubly LinkedList must be used in the above scenario. Circular Doubly LinkedList will store the data in the fashion where it will have the data, the pointer to previous node, the pointer to next node as well as the last node will be pointing to the first node and the first node's prev will point to the last node's next pointer. Hence that will mean that the concatenation of the two lists will require a constant time ,i.e., O(1) time.