

# Rajalakshmi Engineering College

Name: Praveen Kumar  
Email: 240801247@rajalakshmi.edu.in  
Roll no: 240801247  
Phone: 7550385160  
Branch: REC  
Department: I ECE AF  
Batch: 2028  
Degree: B.E - ECE

Scan to verify results



## NeoColab\_REC\_CS23231\_DATA STRUCTURES

### REC\_DS using C\_Week 2\_COD\_Question 2

Attempt : 1  
Total Mark : 10  
Marks Obtained : 10

#### Section 1 : Coding

##### 1. Problem Statement

Moniksha, a chess coach organizing a tournament, needs a program to manage participant IDs efficiently. The program maintains a doubly linked list of IDs and offers two functions: Append to add IDs as students register, and Print Maximum ID to identify the highest ID for administrative tasks.

This tool streamlines tournament organization, allowing Moniksha to focus on coaching her students effectively.

##### ***Input Format***

The first line consists of an integer  $n$ , representing the number of participant IDs to be added.

The second line consists of  $n$  space-separated integers representing the participant IDs.

### **Output Format**

The output displays a single integer, representing the maximum participant ID.

If the list is empty, the output prints "Empty list!".

Refer to the sample output for the formatting specifications.

### **Sample Test Case**

Input: 3

163 137 155

Output: 163

### **Answer**

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

```
typedef struct ID{
    int value;
    struct ID* next;
    struct ID* prev;
}Node;
```

```
void Append(Node** head, int value) {
    Node* node = (Node*) malloc(sizeof(Node));
    node->value = value;
    node->next = NULL;
    node->prev = NULL;
```

```
    if(*head == NULL) {
        *head = node;
        return;
    }
```

```
    Node* temp = *head;
    while(temp->next != NULL) {
        temp = temp->next;
    }
```

```
    temp->next = node;
    node->prev = temp;
```

```
}
```

```
void printMaximum(Node* head) {  
    int max = head->value;
```

```
    while(head != NULL) {  
        if(head->value > max) {  
            max = head->value;  
        }  
        head = head->next;
```

```
    }  
    printf("%d\n", max);  
}
```

```
int main() {  
    Node* head = NULL;  
    int n, m;  
    scanf("%d", &n);  
    if(n < 1) {  
        printf("Empty List!\n");  
        return -1;  
    }  
    for(int i = 0; i < n; i++) {  
        scanf("%d", &m);  
        Append(&head, m);  
    }  
    printMaximum(head);  
}
```

**Status :** Correct

**Marks :** 10/10