2023

LAPCA SCORE REPORT

PES1UG21C454 - PRIYANSHU AGARWAL 2022 Batch,PES University.c

Plagiarism Percentage: 78.45%

Plagiarised File: PES1UG21CS005 - AADITHYA H RAO 2022 Batch,PES University.c

Plagiarised Code:

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

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


node* createLinkedList(int n, int *arr){
    node *head = NULL, *iter = NULL, *newNode = NULL;
    node* dummyhead = (node*)malloc(sizeof(node));
    dummyhead->next = NULL;
    dummyhead->data = 0;
    head = dummyhead;
    iter = head;

    for(int i = 0; i < n; i++){
        newNode = (node*) malloc(sizeof(node  ));
        newNode->data = arr[i];
        newNode->next =  NULL    ;
        iter->next = newNode;
        iter = iter->next;
    }

    return head->next;
}


void insert(node *head, int pos, int data)
{
 node* n=     (node*)malloc(sizeof(node ));
 n->data=data;
 n->next=     NULL;
          if(head==NULL  || pos==0)
 {
  n->next=head;
  head=n;
           return;
 }
 node* temp=head;
 for(      int      i=0;i<pos-1 && temp->next!=NULL;i++)
 {
  temp=temp->next;
  }
  n->next=temp->next;
  temp->next=n;

}


void delete(node *head)
{
  if(head==      NULL      )
  {
   return;
  }
  node*      p=head;
  node*      t;
  while(p!=NULL && p->next!=NULL)
  {
   t=p->      next;
   p->next=t->next;
   p       =t->next;
   free(t);
  }

}


int countNodes(node *head)
{
  int c=0;
  node* t=head;
  while(t!=NULL)
  {
   c       ++;
   t=t->next;
  }
  return c;
}


void printList(node *head){
    if(head == NULL   ){
        printf("List empty!\n");
    }
    node *iter = head;
           while(iter != NULL){
        printf        ("%d ", iter->data);
        iter = iter->next;
    }
    printf("\n");
}


int main(){
    int no_of_nodes;
    scanf("%d",&no_of_nodes);

    int *arr = (int *)malloc(no_of_nodes*sizeof(int));
    for(int i =0;i<no_of_nodes ;i++){
        scanf("%d",&arr[i]);
    }

    node *head = createLinkedList (no_of_nodes, arr);

    int testcases;
    scanf("%d",&testcases);
    int position, data;

    for(int i = 0; i < testcases; i++){
        int operation;
        scanf("%d", &operation);
        switch(operation){
            case 1:
                // Insert at the given position
                scanf("%d %d", &position, &data);
                insert(head, position, data);
                break;
            case 2:
                // Delete alternate nodes, starting from the node at index position 1
                delete(head);
                break;
            case 3:
                // Count the number of nodes
                no_of_nodes = countNodes(head);
                printf(" %d \n ", no_of_nodes);
                break ;
            case 4 :
                // Print the list
                printList( head);
                break;
        }
    }
    return 0;
}
```

PES1UG21CS004 - A.NITHIN 2022 Batch,PES University.c

Plagiarism Percentage: 82.40%

Plagiarised File: PES1UG21CS005 - AADITHYA H RAO 2022 Batch,PES University.c

Plagiarised Code:

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

typedef struct node
```

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


node* createLinkedList(int n, int *arr){
    node *head = NULL, *iter = NULL, *newNode = NULL;

    node* dummyhead = (node*)malloc(sizeof(node));
    dummyhead->next = NULL;
    dummyhead->data = 0;
    head = dummyhead;
    iter = head;

    for(int i = 0; i < n; i++){
        newNode = (node*)  malloc(sizeof(node  ));
        newNode->data = arr[i];
        newNode->next =   NULL     ;
        iter->next = newNode;
        iter = iter->next;
    }

    return head->next;
}


void insert(node *head, int pos, int data){
    node*temp=head;

    for(        int         i=0;i<pos-1&&temp!=NULL;i++)

                temp=temp->next;
        if(temp==NULL    )


       { node*ele=(node*)malloc(sizeof(no ));
        ele->data=data;

         ele->next=temp->next;
        temp->next=ele;
        }
}



void delete(node *head){



}
```

```
int countNodes(node *head){
    int count=0;
    node*temp=head;
     while(temp!=NULL)
    {
        temp=temp->next;
        count++;
    }
        return count;

}


void printList(node *head){
    if(head == NULL ){
        printf("List empty!\n");
    }
    node *iter = head;
    while(iter != NULL){
        printf("%d ", iter->data);
        iter = iter->next;
    }
    printf("\n");
}


int main(){

    int no_of_nodes;
    scanf("%d",&no_of_nodes);

    int *arr = (int *)malloc(no_of_nodes*sizeof(int));
    for(int i =0;i<no_of_nodes ;i++){
        scanf("%d",&arr[i]);
    }

    node *head = createLinkedList (no_of_nodes, arr);

    int testcases;
    scanf("%d ",&testcases);
    int position, data;

    for(int i = 0; i < testcases; i++){
        int operation;
        scanf("%d", &operation);
        switch(operation){
            case 1:
            // Insert at the given position
            scanf("%d %d", &position, &data);
            insert(head, position, data);
            break;
          case 2:
            // Delete alternate nodes, starting from the node at index position 1
            delete(head);
            break;
                case 3:
            // Count the number of nodes
            no_of_nodes =       countNodes(head);
            printf("           %d \n       ", no_of_nodes);
            break          ;
         case 4          :
            // Print the list
            printList(          head);
            break;
        }
    }
    return 0;
}
```

pes1ug21cs021 - abhi ram.c

Plagiarism Percentage: 81.83%

Plagiarised File: PES1UG21CS006 - Aaditya(1).c

Plagiarised Code:

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

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


node* createLinkedList(int n, int *arr){
    node *head = NULL, *iter = NULL, *newNode = NULL;

    node* dummyhead = (node*)malloc(sizeof(node));
    dummyhead->next = NULL;
    dummyhead->data = 0;
    head = dummyhead;
    iter = head;

    for(int i = 0; i < n; i++){
        newNode = (node*)  malloc(sizeof(node  ));
        newNode->data = arr[i];
        newNode->next =  NULL     ;
        iter->next = newNode;
        iter = iter->next;
    }

    return head->next;
}
```

```
void insert(node *head, int pos, int data){
    // TODO: Insert the data at given pos in the linked list
    node* temp =      (node*)malloc(sizeof(node));
    temp ->data = data;
    temp ->next = NULL;

    if(head == NULL   ){
    head =  temp;
    }


             node *a = head;
     node *b = NULL;
             int c=0        ;
     while(a!=        NULL && c<   =pos){
         b = a;
         a = a->next;
         c++;

           }
               if(          b == NULL){
       temp->next = head;
       head =  temp;

     }
     else{
     b->next =  temp;
     temp->next = a;
     }
}


void delete(node *head){
    // TODO: Delete alternate nodes starting at index 1
    int i=0;
    node *a = head;
             node *b = NULL;
    while        (a != NULL){
       a = b          ;
       a = a->next;
       i++;
       if(i%2 != 0        ){
          b->next = a->next;
          free(a);
          i          ++;
                    a = b->next;
          }

    }
```

```
}


int countNodes(node *head){
    // TODO: Count the number of nodes in the given list
            node *a = head;
    int i=0;
    while(a !        = NULL){
      a = a->next;
      i++;
    }
    return i;

}


void        printList(node *head){
    if(head ==       NULL    ){
        printf("List empty!\n");
    }
    node *iter = head;
    while(iter !        = NULL){
        printf        ("%d ", iter->data);
        iter = iter->next;
    }
    printf("\n");
}


int main(){

    int no_of_nodes;
    scanf("%d",&no_of_nodes);

    int *arr = (int *)malloc(no_of_nodes*sizeof(int));
    for(int i        =0;i<no_of_nodes ;i++){
        scanf("%d",&arr[i]);
    }

    node *head =    createLinkedList   (no_of_nodes, arr);

    int testcases;
            scanf("%d    ",&testcases);
    int position, data;

    for(int         i = 0; i         < testcases; i++){
        int operation;
                scanf("%d",    &operation);
        switch(operation){
                    case       1:
                // Insert at the given position
                scanf("%d %d", &position, &data);
                insert(head, position, data);
                break;
            case 2:
                // Delete alternate nodes, starting from the node at index position 1
                delete(head);
                break;
                    case 3:
                // Count the number of nodes
                no_of_nodes =      countNodes(head);
                printf("          %d \n       ", no_of_nodes);
                break          ;
            case 4          :
                // Print the list
                printList(           head);
                break;
        }
    }
    return 0;
}
```

PES1UG21CS006 - Aaditya(1).c

Plagiarism Percentage: 87.36%

Plagiarised File: PES1UG21CS006 - Aaditya.c

Plagiarised Code:

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

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


node* createLinkedList(int n, int *arr){
    node *head = NULL, *iter = NULL, *newNode = NULL;

    node* dummyhead = (node*)malloc(sizeof(node));
    dummyhead->next = NULL;
    dummyhead->data = 0;
    head = dummyhead;
    iter = head;

    for(int i = 0; i < n; i++){
        newNode = (node*)  malloc(sizeof(node  ));
        newNode->data = arr[i];
        newNode->next =   NULL    ;
        iter->next = newNode;
        iter = iter->next;
    }

    return head->next;
}


void insert(node *head, int pos, int data){
    // TODO: Insert the data at given pos in the linked list
    node *p=head;
    node *q=NULL;
    node        *temp=(node*)malloc(sizeof(nc ));
    temp->data= data;
    temp->next=NULL;
    if (pos==0 || head==NULL)
    {
        temp->next=head;
        head=temp;
    }
    else
    {
        for (        int i=0;i        <pos;i++)
        {q=p;p=p->next;}
        q->next=temp;
        temp->next=p;
    }


}


void delete(node *head){
    // TODO: Delete alternate nodes starting at index 1
    node *t;
    node *p=head;
    while(p)
    {
        t=p->next;
        if (p->next)
                p->next=p->next->next;
        else p->next==NULL;
        free(t);
        p=p->next;

    }
}


int countNodes(node *head){
    // TODO: Count the number of nodes in the given list
    int count=0;
    node *p=head;
    if (head=NULL)
    {        return 0        ;}
    else
    {
        while(p!=NULL)
        {
                p=p->next;
                count++;
        }
    }
    return count;



}


void printList(node *head){
    if(head == NULL){
        printf("List empty!\n");
    }
    node *iter = head;
    while(iter != NULL){
        printf("%d ", iter->data);
        iter = iter->next;
    }
    printf("\n");
}


int main(){

    int no_of_nodes;
    scanf("%d",&no_of_nodes);

    int *arr = (int *)malloc(no_of_nodes*sizeof(int));
    for(int i =0;i<no_of_nodes ;i++){
        scanf("%d",&arr[i]);
    }

    node *head = createLinkedList (no_of_nodes, arr);

    int testcases;
           scanf("%d ",&testcases);
    int position, data;

    for(int i = 0; i < testcases; i++){
        int operation;
             scanf("%d", &operation);
        switch(operation){
                 case 1:
                // Insert at the given position
                scanf("%d %d", &position, &data);
                insert(head, position, data);
                break;
            case 2:
                // Delete alternate nodes, starting from the node at index position 1
                delete(head);
                break;
                    case 3:
                // Count the number of nodes
                no_of_nodes =       countNodes(head);
                printf("         %d \n       ", no_of_nodes);
                break         ;
            case 4          :
                // Print the list
                printList(         head);
                break;
        }
    }
    return 0;
}
```

PES1UG21CS006 - Aaditya.c

Plagiarism Percentage: 87.36%

Plagiarised File: PES1UG21CS006 - Aaditya(1).c

Plagiarised Code:

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

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


node* createLinkedList(int n, int *arr){
    node *head = NULL, *iter = NULL, *newNode = NULL;

    node* dummyhead = (node*)malloc(sizeof(node));
    dummyhead->next = NULL;
    dummyhead->data = 0;
    head = dummyhead;
    iter = head;

    for(int i = 0; i < n; i++){
        newNode = (node*)  malloc(sizeof(node  ));
        newNode->data = arr[i];
        newNode->next =   NULL     ;
        iter->next = newNode;
        iter = iter->next;
    }

    return head->next;
}


void insert(node *head, int pos, int data){
    // TODO: Insert the data at given pos in the linked list
    node *p=head;
    node *q=NULL;
    node        *temp=(node*)malloc(sizeof(nc ));
    temp->data= data;
    temp->next=NULL;
    if (pos==0 || head==NULL)
    {
        temp->next=head;
        head=temp;
    }
    else
    {
        for (        int i=0;i         <pos;i++)
        {q=p;p=p->next;}
        q->next=temp;
        temp->next=p;
    }


}


void delete(node *head){
    // TODO: Delete alternate nodes starting at index 1
    node *t;
    node *p=head;
    while(p)
    {
        t=p->next;
        if (p->next)
                p->next=p->next->next;
        else p->next==NULL;
        free(t);
        p=p->next;

    }
}


int countNodes(node *head){
    // TODO: Count the number of nodes in the given list
    int count=0;
    node *p=head;
    if (head=NULL)
    {       return 0        ;}
    else
    {
        while(p!=NULL)
        {
                p=p->next;
                count++;
        }
    }
    return count;



}


void    printList(node *head){
    if(head == NULL){
        printf("List empty!\n");
    }
    node *iter = head;
    while(iter !=    NULL){
        printf    ("%d ", iter->data);
        iter = iter->next;
    }
    printf("\n");
}


int main(){

    int no_of_nodes;
    scanf("%d",&no_of_nodes);

    int *arr = (int *)malloc(no_of_nodes*sizeof(int));
    for(int i    =0;i<no_of_nodes ;i++){
        scanf("%d",&arr[i]);
    }

    node *head =  createLinkedList  (no_of_nodes, arr);

    int testcases;
           scanf("%d   ",&testcases);
    int position, data;

    for(int    i = 0; i    < testcases; i++){
        int operation;
             scanf("%d",   &operation);
        switch(operation){
                case    1:
                // Insert at the given position
                scanf("%d %d", &position, &data);
                insert(head, position, data);
                break;
            case 2:
                // Delete alternate nodes, starting from the node at index position 1
                delete(head);
                break;
                    case 3:
                // Count the number of nodes
                no_of_nodes =       countNodes(head);
                printf("         %d \n       ", no_of_nodes);
                break          ;
            case 4           :
                // Print the list
                printList(          head);
                break;
        }
    }
    return 0;
}
```

PES1UG19CS359 - Rachappa Biradar.c

Plagiarism Percentage: 78.45%

Plagiarised File: pes1ug21cs021 - abhi ram.c

Plagiarised Code:

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

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


node* createLinkedList(int n, int *arr){
    node *head = NULL, *iter = NULL, *newNode = NULL;

    node* dummyhead = (node*)malloc(sizeof(node));
    dummyhead->next = NULL;
    dummyhead->data = 0;
    head = dummyhead;
    iter = head;

    for(int i = 0; i < n; i++){
        newNode = (node*)  malloc(sizeof(node  ));
        newNode->data = arr[i];
        newNode->next =  NULL     ;
        iter->next = newNode;
        iter = iter->next;
    }

    return head->next;
}


void insert(node *head, int pos, int data){
    // TODO: Insert the data at given pos in the linked list
    node *temp =head;
    node *newnode =   (node*)malloc(sizeof(node));
    newnode->data = data;
    newnode->next =NULL;
    int i = 0;
    node *prev =NULL;
    while        ((temp !=NULL) && (i<pos-1))
    {
        prev = temp       ;
        i++;
        temp = temp->next;
    }
    if(prev ==NULL    )
    {
        if(pos ==0        )
        {
        newnode->next = temp;
        head = newnode;
                 return;
        }
     }
     if(temp == NULL)
     {
        return          ;
      }
      else
      {
        newnode->next = temp->next;
                 temp       ->next = newnode;
       }






    }




void delete(node *head){
    // TODO: Delete alternate nodes starting at index 1

    node *prev =head;
    node *cur = head   ->        next;
    node *nex = NULL   ;
    while(cur &&       prev)
    {
    nex = cur->next;
    prev        ->next =        nex;
    free(cur);
             if(nex==NULL)
    {
    return;


    }
    cur= nex->next;
    prev =         nex;



    }

}


int countNodes(node *head){
    // TODO: Count the number of nodes in the given list
    int i  =0;
    node *temp =head;
    while(temp!=NULL)
    {
    i         ++;
    temp  = temp->next;
    }
    return i ;
}


void          printList(node *head){
    if(head ==        NULL      ){
        printf("List empty!\n");
    }
    node *iter = head    ;
    while(iter != NULL){
        printf           ("%d ", iter->data);
        iter = iter->next;
    }
    printf("\n");
}


int main(){

    int no_of_nodes;
    scanf("%d",&no_of_nodes);
    int *arr = (int *)malloc(no_of_nodes*sizeof(int));
    for(int i =0;i<no_of_nodes ;i++){
        scanf("%d",&arr[i]);
    }

    node *head = createLinkedList (no_of_nodes, arr);

    int testcases;
        scanf("%d ",&testcases);
    int position, data;

    for(int i = 0; i < testcases; i++){
        int operation;
            scanf("%d", &operation);
        switch(operation){
            case 1:
                // Insert at the given position
                scanf("%d %d", &position, &data);
                insert(head, position, data);
                break;
            case 2:
                // Delete alternate nodes, starting from the node at index position 1
                delete(head);
                break;
            case 3:
                // Count the number of nodes
                no_of_nodes = countNodes(head);
                printf(" %d \n ", no_of_nodes);
                break ;
            case 4 :
                // Print the list
                printList( head);
                break;
        }
    }
    return 0;
}
```

_PES1UG21CS730 - Yashas Shettar.c

Plagiarism Percentage: 51.62%

Plagiarised File: pes1ug21cs021 - abhi ram.c

Plagiarised Code:

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

typedef struct cache
{
    int capacity;
    // declare some additional data-structure(s) here if necessary
}   cache_t;



void access(cache_t* cache,unsigned data)
{
    // TODO
}


int LRU(cache_t* cache)
{
    // TODO
}


cache_t * init_cache( int capacity)
{
    cache_t* c = (cache_t*)malloc(sizeof(cache_t));  // 1st line
    c->capacity = capacity;                           // 2nd line
    // TODO
    // if cache_t structure was added with more data-structure(s), initialize them here
        return c;
}


void free_cache(cache_t* c)
{
    // TODO
    // if cache_t structure was added with more data-structure(s), free the heap space it consumes, if any, here
        free(c) ; // last line
}


// Driver program to test above functions
int main()
{
    int capacity;
    scanf("%d", &capacity);
    int T ;
    scanf("%d", &T);
    char op[10];
    int data;
    cache_t* cache = init_cache(capacity);

    while(T-- )
    {
        scanf("%s" , op);
        if(!strcmp(op, "access"))
        {
            scanf("%d", &data);
            access(cache, data);
        }
        else
        {
                    printf("%d\n    ", LRU(cache));
        }
    }

    free_cache(cache);

      return 0;
}
```

PES1UG20CS041 - Ananth k.c

Plagiarism Percentage: 80.49%

Plagiarised File: pes1ug21cs021 - abhi ram.c

Plagiarised Code:

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

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


node* createLinkedList(int n, int *arr){
    node *head = NULL, *iter = NULL, *newNode = NULL;

    node* dummyhead = (node*)malloc(sizeof(node));
    dummyhead->next = NULL;
    dummyhead->data = 0;
    head = dummyhead;
    iter = head;

    for(int i = 0; i < n; i++){
        newNode = (node*)  malloc(sizeof(node  ));
        newNode->data = arr[i];
        newNode->next =  NULL    ;
        iter->next = newNode;
        iter = iter->next;
    }

    return head->next;
}


int countNodes(node *head){
    // TODO: Count the number of nodes in the given list
    int i=      1;
    node     *temp=head;
    while(temp   ->     next!=NULL)
    {
       temp=temp->next;
       i++;
    }
    if(temp==NULL  )
    return 0;
    else
    return      i;

}


void insert(node *head, int pos, int data)
    // TODO: Insert the data at given pos in the linked list
{
        node* curr=head;
        node*new=head  ->next;
        if(pos<0)
        return;
        node* temp=(node*)malloc(sizeof(n));
        temp->data=data;
        temp->next=NULL;
        int j=       1;
        if(pos==0      )
        {
        temp->next=head;
        head=temp;
        }
        else
        {
            while(     j<pos)
        {
        curr=new;
        new=new->next;
        j+        +;
        }
        curr->next      =temp;
        temp->next=new;

 }

}

void delete(node *head){
    // TODO: Delete alternate nodes starting at index 1
        node*curr=head;
            node*new=head->next;

        while(curr->next!     =NULL)
        {
        curr->next=new->next;
                  curr=new;
        new=      new->next;
        }


}



void printList(node *head){
    if(head ==      NULL    ){
        printf("List empty!\n");
    }
    node *iter = head;
    while(iter !=       NULL){
        printf        ("%d ", iter->data);
        iter = iter->next;
    }
    printf("\n");
}


int main(){

    int no_of_nodes;
    scanf("%d",&no_of_nodes);

    int *arr = (int *)malloc(no_of_nodes*sizeof(int));
    for(int i          =0;i<no_of_nodes ;i++){
        scanf("%d",&arr[i]);
    }

    node *head =     createLinkedList  (no_of_nodes, arr);

    int testcases;
              scanf("%d     ",&testcases);
    int position, data;

    for(int         i = 0; i         < testcases; i++){
        int operation;
                  scanf("%d",     &operation);
        switch(operation){
                      case        1:
              // Insert at the given position
              scanf("%d %d", &position, &data);
              insert(head, position, data);
              break;
           case 2:
              // Delete alternate nodes, starting from the node at index position 1
              delete(head);
              break;
```

```
            case 3:
                // Count the number of nodes
                no_of_nodes =   countNodes(head);
                printf("  %d \n  ", no_of_nodes);
                break  ;
            case 4  :
                // Print the list
                printList(  head);
                break;
        }
    }
    return 0;
}
```

PES1UG21CS007 - AAKASH V 2022 Batch,PES University.c

Plagiarism Percentage: 87.57%

Plagiarised File: PES1UG21CS007 - AAKASH V 2022 Batch,PES University(1).c

Plagiarised Code:

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

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


node* createLinkedList(int n, int *arr){
    node *head = NULL, *iter = NULL, *newNode = NULL;

    node* dummyhead = (node*)malloc(sizeof(node));
    dummyhead->next = NULL;
    dummyhead->data = 0;
    head = dummyhead;
    iter = head;

    for(int i = 0; i < n; i++){
        newNode = (node*)  malloc(sizeof(node  ));
        newNode->data = arr[i];
        newNode->next =  NULL  ;
        iter->next = newNode;
        iter = iter->next;
    }

    return head->next;
}
```

```
void insert(node *head, int pos, int data){
   // TODO: Insert the data at given pos in the linked list

       node* temp = (node*)  malloc(sizeof(node));
       temp->next= NULL;
                temp->data = data;

       for(int i = 1; i < pos; i++){
             if(head==NULL) return;
             if(head->data != data)
             head = head->next;
       }
       if(head) {
             temp->next = head->next;
             head->next = temp;
       }

}


void delete(node *head){
   // TODO: Delete alternate nodes starting at index 1
/*
       node*temp = head->next;
       free(head);
       head = temp;*/
       node*temp;
       node*curr = head;
       while(curr){
             temp = curr->next;
             if(curr->next)
                   curr->next = curr->next->next;
             else curr->next = NULL;
             free(temp);
             curr = curr->next;

       }

}


int countNodes(node *head){
   // TODO: Count the number of nodes in the given list

       int count = 0;

       node* temp = head;
                while(      temp){
             temp = temp->next;
             count++;
       }
```

```
        return count;

}


void        printList(node *head){
    if(head ==       NULL    ){
        printf("List empty!\n");
    }
    node *iter = head;
    while(iter !=        NULL){
        printf        ("%d ", iter->data);
        iter = iter->next;
    }
    printf("\n");
}


int main(){

    int no_of_nodes;
    scanf("%d",&no_of_nodes);

    int *arr = (int *)malloc(no_of_nodes*sizeof(int));
    for(int i          =0;i<no_of_nodes ;i++){
        scanf("%d",&arr[i]);
    }

    node *head =    createLinkedList    (no_of_nodes, arr);

    int testcases;
                scanf("%d    ",&testcases);
    int position, data;

    for(int          i = 0; i          < testcases; i++){
        int operation;
                 scanf("%d",     &operation);
        switch(operation){
                      case       1:
                // Insert at the given position
                scanf("%d %d", &position, &data);
                insert(head, position, data);
                break;
            case 2:
                // Delete alternate nodes, starting from the node at index position 1
                delete(head);
                break;
                      case 3:
                // Count the number of nodes
                no_of_nodes =       countNodes(head);
                printf("          %d \n      ", no_of_nodes);
                break         ;
            case 4          :
```

```
            // Print the list
            printList (head);
            break;
        }
    }
    return 0;
}
```

PES1UG21CS007 - AAKASH V 2022 Batch,PES University(1).c

Plagiarism Percentage: 87.57%

Plagiarised File: PES1UG21CS007 - AAKASH V 2022 Batch,PES University.c

Plagiarised Code:

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

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


node* createLinkedList(int n, int *arr){
    node *head = NULL, *iter = NULL, *newNode = NULL;

    node* dummyhead = (node*)malloc(sizeof(node));
    dummyhead->next = NULL;
    dummyhead->data = 0;
    head = dummyhead;
    iter = head;

    for(int i = 0; i < n; i++){
        newNode = (node*) malloc(sizeof(node ));
        newNode->data = arr[i];
        newNode->next = NULL ;
        iter->next = newNode;
        iter = iter->next;
    }

    return head->next;
}


void insert(node *head, int pos, int data){
    // TODO: Insert the data at given pos in the linked list

        node* temp = (node*) malloc(sizeof(node));
        temp->next= NULL;
                temp->data = data;
        for(int i = 1; i < pos; i++){
                if(head==NULL) return;
                if(head->data != data)
                head = head->next;
        }
        if(head) {
                temp->next = head->next;
                head->next = temp;
        }

}


void delete(node *head){
    // TODO: Delete alternate nodes starting at index 1
/*
        node*temp = head->next;
        free(head);
        head = temp;*/
        node*temp;
        node*curr = head;
        while(curr){
                temp = curr->next;
                if(curr->next)
                        curr->next = curr->next->next;
                else curr->next = NULL;
                free(temp);
                curr = curr->next;

        }

}


int countNodes(node *head){
    // TODO: Count the number of nodes in the given list

        int count = 0;

        node* temp = head;
                while(    temp){
                temp = temp->next;
                count++;
        }
        return count;

}


void      printList(node *head){
    if(head ==      NULL    ){
        printf("List empty!\n");
    }
    node *iter = head;
    while(iter != NULL){
        printf("%d ", iter->data);
        iter = iter->next;
    }
    printf("\n");
}


int main(){

    int no_of_nodes;
    scanf("%d",&no_of_nodes);

    int *arr = (int *)malloc(no_of_nodes*sizeof(int));
    for(int i=0;i<no_of_nodes;i++){
        scanf("%d",&arr[i]);
    }

    node *head = createLinkedList(no_of_nodes, arr);

    int testcases;
    scanf("%d",&testcases);
    int position, data;

    for(int i = 0; i < testcases; i++){
        int operation;
        scanf("%d", &operation);
        switch(operation){
            case 1:
                // Insert at the given position
                scanf("%d %d", &position, &data);
                insert(head, position, data);
                break;
            case 2:
                // Delete alternate nodes, starting from the node at index position 1
                delete(head);
                break;
            case 3:
                // Count the number of nodes
                no_of_nodes = countNodes(head);
                printf("%d \n", no_of_nodes);
                break;
            case 4:
                // Print the list
                printList(head);
                break;
        }
    }
    return 0;
}
```

PES1UG21CS005 - AADITHYA H RAO 2022 Batch,PES University.c

Plagiarism Percentage: 81.69%

Plagiarised File: PES1UG21C454 - PRIYANSHU AGARWAL 2022 Batch,PES Universit

Plagiarised Code:

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

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


node* createLinkedList(int n, int *arr){
    node *head = NULL, *iter = NULL, *newNode = NULL;

    node* dummyhead = (node*)malloc(sizeof(node));
    dummyhead->next = NULL;
    dummyhead->data = 0;
    head = dummyhead;
    iter = head;

    for(int i = 0; i < n; i++){
        newNode = (node*)  malloc(sizeof(node  ));
        newNode->data = arr[i];
        newNode->next =  NULL     ;
        iter->next = newNode;
        iter = iter->next;
    }

    return head->next;
}


void insert(node *head, int pos, int data){
    node* temp = head;



    for(      int       i=0; i<pos-1 && temp!=NULL; i++)
        temp = temp->next;

    if(temp==NULL)
        return;

    node* ele       = (node*)malloc(sizeof(nod ));
    ele->data = data;

    ele->next = temp->next;
    temp->next = ele;



}


void delete(node *head){
        node* temp = head;
    node* prev = NULL ;

    while(temp!=NULL && temp->next!=NULL)
    {
        prev = temp ;
        temp = temp->next;

        prev->next = temp->next;
        free(temp);
        temp = prev->next;
    }


}


int countNodes(node *head){
    int count = 0;

    node* temp = head;
    while(temp!=NULL)
    {
        temp = temp-> next;
        count++;
    }

    return count;



}


void printList(node *head){
    if(head == NULL ){
        printf("List empty!\n");
    }
    node *iter = head;
        while(iter != NULL){
        printf ("%d ", iter->data);
        iter = iter->next;
    }
    printf("\n");
}


int main(){

    int no_of_nodes;
    scanf("%d",&no_of_nodes);

    int *arr = (int *)malloc(no_of_nodes*sizeof(int));
    for(int i =0;i<no_of_nodes ;i++){
        scanf("%d",&arr[i]);
    }

    node *head = createLinkedList(no_of_nodes, arr);

    int testcases;
    scanf("%d",&testcases);
    int position, data;

    for(int i = 0; i < testcases; i++){
        int operation;
        scanf("%d", &operation);
        switch(operation){
            case 1:
                // Insert at the given position
                scanf("%d %d", &position, &data);
                insert(head, position, data);
                break;
            case 2:
                // Delete alternate nodes, starting from the node at index position 1
                delete(head);
                break;
            case 3:
                // Count the number of nodes
                no_of_nodes = countNodes(head);
                printf("%d \n", no_of_nodes);
                break;
            case 4:
                // Print the list
                printList(head);
                break;
        }
    }
    return 0;
}
```

PES1ug21cs136 - Bharath Raja.c

Plagiarism Percentage: 79.13%

Plagiarised File: PES1UG21CS006 - Aaditya(1).c

Plagiarised Code:

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

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


node* createLinkedList(int n, int *arr){
    node *head = NULL, *iter = NULL, *newNode = NULL;

    node* dummyhead = (node*)malloc(sizeof(node));
    dummyhead->next = NULL;
    dummyhead->data = 0;
    head = dummyhead;
    iter = head;

    for(int i = 0; i < n; i++){
        newNode = (node*)  malloc(sizeof(node  ));
        newNode->data = arr[i];
        newNode->next =  NULL     ;
        iter->next = newNode;
        iter = iter->next;
    }

    return head->next;
}


void insert(node *head, int pos, int data){
    // TODO: Insert the data at given pos in the linked list
    node* temp =     (node*)malloc(sizeof(node ));
    temp->data = data;
    temp->next =     NULL;
            node* prev = head;
    node* curr = head;
    if(head==NULL  )
        head = temp;
    for(int i=0; i          <pos; i++){
        prev = curr;
        curr = curr->next;
    }
    prev->next = temp;
    temp->next = curr;


}
```

```
void delete(node *head){
    // TODO: Delete alternate nodes starting at index 1
    node* prev = head;
    node* curr        = head->next;
    node*        ahead = head->next->next;
    while(ahead != NULL){
        prev->next =        ahead;
        curr->next        = NULL;
        free(curr);
        curr =         NULL;
        prev = ahead;
        curr = ahead->next;
        ahead = curr->next;
    }
    prev->next = NULL;
    free(curr);
    curr = NULL      ;


}


int countNodes(node *head){
    int count = 0;
    node* temp = head;
    while(temp != NULL){
        temp = temp->next;
        count++;
    }
    return count;


}


void         printList(node *head){
    if(head ==         NULL     ){
        printf("List empty!\n");
    }
    node *iter = head;
            while(iter != NULL){
        printf          ("%d ", iter->data);
        iter = iter->next;
    }
    printf("\n");
}


int main(){

    int no_of_nodes;
    scanf("%d",&no_of_nodes);
    int *arr = (int *)malloc(no_of_nodes*sizeof(int));
    for(int i           =0;i<no_of_nodes ;i++){
       scanf("%d",&arr[i]);
    }

    node *head =  createLinkedList  (no_of_nodes, arr);

    int testcases;
             scanf("%d     ",&testcases);
    int position, data;

    for(int         i = 0; i         < testcases; i++){
       int operation;
               scanf("%d",   &operation);
       switch(operation){
                case      1:
             // Insert at the given position
             scanf("%d %d", &position, &data);
             insert(head, position, data);
             break;
          case 2:
             // Delete alternate nodes, starting from the node at index position 1
             delete(head);
             break;
                case 3:
             // Count the number of nodes
             no_of_nodes =       countNodes(head);
             printf("         %d \n      ", no_of_nodes);
             break        ;
          case 4       :
             // Print the list
             printList(          head);
             break;
       }
    }
    return 0;
 }
```