Assignment 4

DSA LAB

2029196

Adarsh Kumar

1. Create a single linked list and perform following operations:

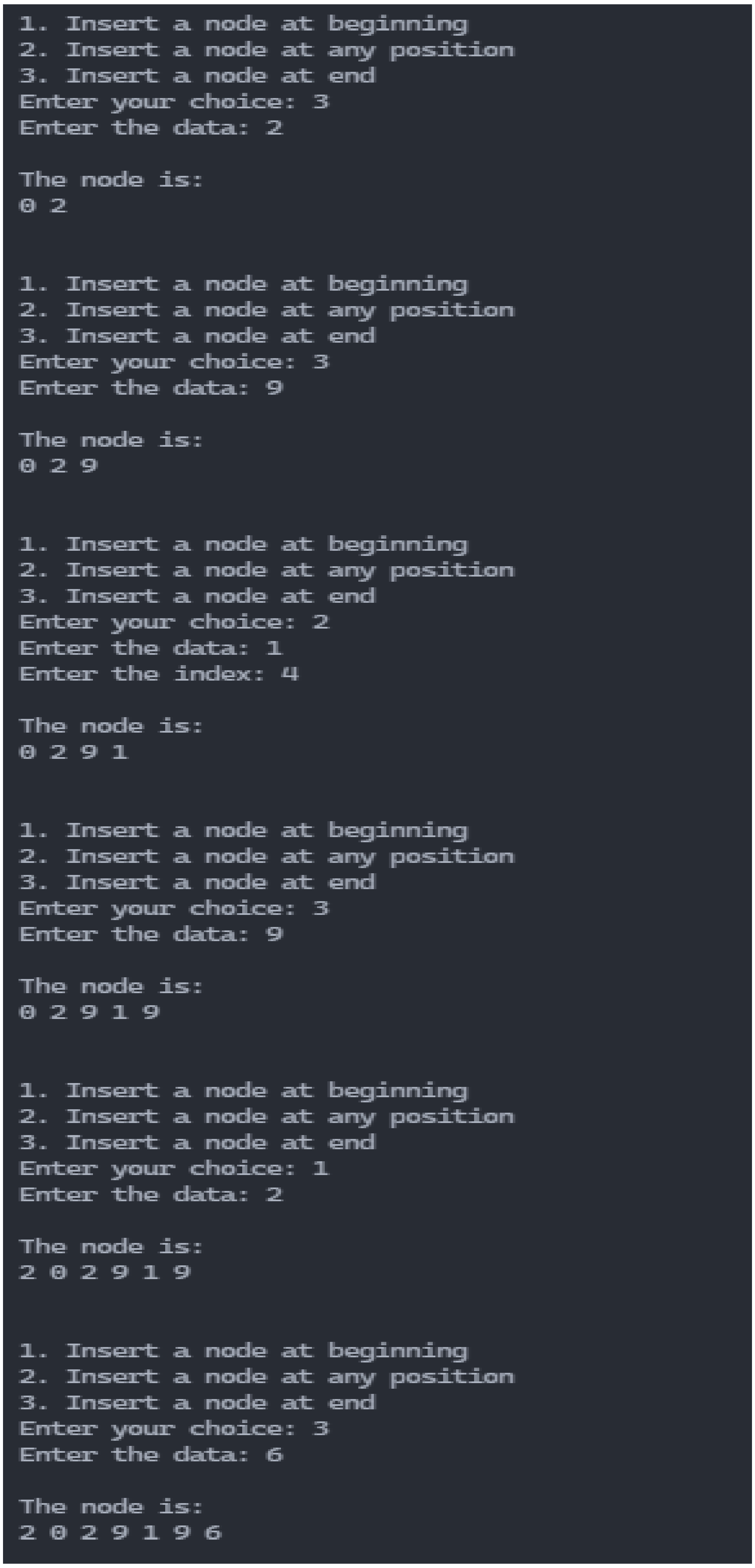
(a) Insert the new list at any position.
(b) Insert the new list in front.
(c) Insert the new list at the end.

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

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

struct node *head = NULL;

void displayList();

void insertAtBegining()
{
    int newData;
    printf("Enter the data: ");
    scanf("%d", &newData);

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

    newNode->data = newData;
    newNode->next = head;
    head = newNode;
}

void insertAtIndex()
{
    int newData, position;
    printf("Enter the data: ");
    scanf("%d", &newData);
    printf("Enter the index: ");
    scanf("%d", &position);

    struct node *ptr = malloc(sizeof(struct node)), *temp = head;
    ptr->data = newData;
    int i;
    if (position == 1)
    {
        ptr->next = temp;
        head = ptr;
        return;
    }
    for (i = 1; i < position - 1; i++)
    {
        temp = temp->next;
    }
    ptr->next = temp->next;
    temp->next = ptr;
}

void insertAtEnd()
{
    int newData;
    printf("Enter the data: ");
    scanf("%d", &newData);

    struct node *newNode = malloc(sizeof(struct node));
    struct node *last = head;
    newNode->data = newData;
    newNode->next = NULL;
    if (head == NULL)
    {
        head = newNode;
        return;
    }
    while (last->next != NULL)
        last = last->next;
    last->next = newNode;
    return;
}
```

```c
void insert()
{
    int ch;
    printf("\n");
    printf("1. Insert a node at beginning\n");
    printf("2. Insert a node at any position\n");
    printf("3. Insert a node at end\n");
    printf("Enter your choice: ");
    scanf("%d", &ch);
    if (ch == 1)
        insertAtBegining();
    else if (ch == 2)
        insertAtIndex();
    else if (ch == 3)
        insertAtEnd();
    else
        printf("Invalid argument");

    displayList();
    printf("\n");
}

void displayList()
{
    struct node *current = malloc(sizeof(struct node));
    current = head;
    printf("\nThe node is:\n");
    while (current != NULL)
    {
        printf("%d ", current->data);
        current = current->next;
    }
    free(current);
    printf("\n");
}

int main()
{
    int n;
    printf("Enter the number of nodes you want to enter: ");
    scanf("%d", &n);

    for (int i = 0; i < n; i++)
```

```
        insert();

    return 0;
}
```

OUTPUT:-



```
1. Insert a node at beginning
2. Insert a node at any position
3. Insert a node at end
Enter your choice: 3
Enter the data: 2

The node is:
0 2


1. Insert a node at beginning
2. Insert a node at any position
3. Insert a node at end
Enter your choice: 3
Enter the data: 9

The node is:
0 2 9


1. Insert a node at beginning
2. Insert a node at any position
3. Insert a node at end
Enter your choice: 2
Enter the data: 1
Enter the index: 4

The node is:
0 2 9 1


1. Insert a node at beginning
2. Insert a node at any position
3. Insert a node at end
Enter your choice: 3
Enter the data: 9

The node is:
0 2 9 1 9


1. Insert a node at beginning
2. Insert a node at any position
3. Insert a node at end
Enter your choice: 1
Enter the data: 2

The node is:
2 0 2 9 1 9


1. Insert a node at beginning
2. Insert a node at any position
3. Insert a node at end
Enter your choice: 3
Enter the data: 6

The node is:
2 0 2 9 1 9 6
```

2. Create a single linked list and perform following operations:

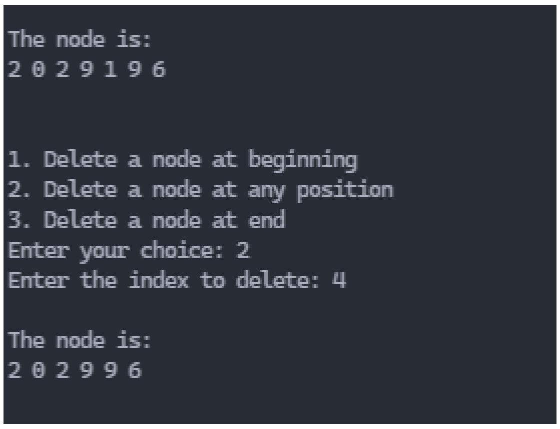
 (a) Delete the list from any postion.
 (b) Delete the list fromthe front.
 (c) Delete the list from the end.

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

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

struct node *head = NULL;

void insert(int newData)
{
    struct node *newNode = malloc(sizeof(struct node));
    struct node *last = head;

    newNode->data = newData;
    newNode->next = NULL;

    if (head == NULL)
    {
        head = newNode;
        return;
    }

    while (last->next != NULL)
        last = last->next;

    last->next = newNode;
    return;
}

void displayList()
{
    struct node *current = malloc(sizeof(struct node));
    current = head;
    printf("\nThe node is:\n");
    while (current != NULL)
    {
        printf("%d ", current->data);
        current = current->next;
    }
    free(current);
    printf("\n\n");
}

void deleteAtBegining()
{
    struct node *current;
    if (head == NULL)
        printf("List is empty ERROR Deleting");
    else
    {
        current = head;
        head = head->next;
        free(current);
    }
}

void deleteAtIndex()
{
    int position;
    printf("Enter the index to delete: ");
    scanf("%d", &position);

    struct node *temp = head;

    if (position == 0)
    {
        head = temp->next;
        free(temp);
        return;
    }

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

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

    struct node *next = temp->next->next;
    free(temp->next);
    temp->next = next;
```

```c
}

void deleteAtEnd()
{
    struct node *current, *previous;
    if (head == NULL)
        printf("List is empty ERROR Deleting");
    else
    {
        current = head;
        while (current->next != 0)
        {
            previous = current;
            current = current->next;
        }
        free(current);
        previous->next = 0;
    }
}

void delete ()
{
    int ch;
    printf("\n");
    printf("1. Delete a node at beginning\n");
    printf("2. Delete a node at any position\n");
    printf("3. Delete a node at end\n");
    printf("Enter your choice: ");
    scanf("%d", &ch);
    if (ch == 1)
        deleteAtBegining();
    else if (ch == 2)
        deleteAtIndex();
    else if (ch == 3)
        deleteAtEnd();
    else
        printf("Invalid argument");

    displayList();
    printf("\n");
}

int main()
```

```c
{
    insert(2);
    insert(0);
    insert(2);
    insert(9);
    insert(1);
    insert(9);
    insert(6);

    displayList();
    delete ();
}
```

OUTPUT:-



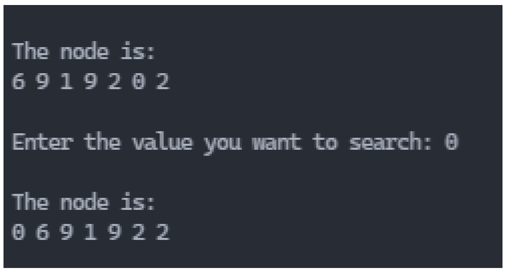
```
The node is:
2 0 2 9 1 9 6


1. Delete a node at beginning
2. Delete a node at any position
3. Delete a node at end
Enter your choice: 2
Enter the index to delete: 4

The node is:
2 0 2 9 9 6
```

3. WAP to search an element in a simple linked list, if found delete that node and insert that node at beginning. Otherwise display an appropriate message.

```c
// WAP to search an element in a simple linked list, if found delete
 that node node and insert that node at beginning.Otherwise display
an appropriate message.

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

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

struct node *head = NULL, *tail = NULL;

void insert(int newData)
```

```c
{
    struct node *newNode = malloc(sizeof(struct node));

    newNode->data = newData;
    newNode->next = head;
    head = newNode;
}

bool search(int n)
{
    struct node *current = head;
    while (current != NULL)
    {
        if (current->data == n)
        {
            return true;
        }
        current = current->next;
    }
    return false;
}

void deletionAtKey(int key)
{
    struct node *current = head, *previous;

    if (current != NULL && current->data == key)
    {
        head = current->next;
        free(current);
        return;
    }

    while (current != NULL && current->data != key)
    {
        previous = current;
        current = current->next;
    }

    if (current == NULL)
        return;

    previous->next = current->next;
    free(current);
}

void insertAtBegining(int newData)
{
    struct node *newNode = malloc(sizeof(struct node));

    newNode->data = newData;
    newNode->next = head;
    head = newNode;
}

void operationOnList()
{
    int num;
    printf("Enter the value you want to search: ");
    scanf("%d", &num);

    search(num) ? deletionAtKey(num) : printf("The value is not pres
ent\n\n");

    insertAtBegining(num);
}

void displayList()
{
    struct node *current = malloc(sizeof(struct node));
    current = head;
    printf("\nThe node is:\n");
    while (current != NULL)
    {
        printf("%d ", current->data);
        current = current->next;
    }
    free(current);
    printf("\n\n");
}

int main()
{
    insert(2);
    insert(0);
    insert(2);
    insert(9);
    insert(1);
    insert(9);
    insert(6);

    displayList();
    operationOnList();
    displayList();

    return 0;
}
```

OUTPUT:-



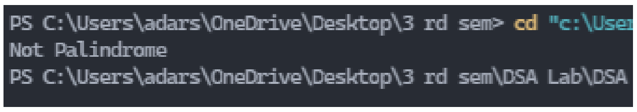
```
The node is:
6 9 1 9 2 0 2

Enter the value you want to search: 0

The node is:
0 6 9 1 9 2 2
```

```
4. WAP to check whether a singly linked list is a palindrome or not.
```

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

struct node
{
    char data;
    struct node *next;
};

bool isPalindrome(struct node *head, int count)
{
    struct node *front, *rear;
    int i = 0;
    while (i != count / 2)
    {
        front = rear = head;
        for (int j = 0; j < i; j++)
            front = front->next;
        for (int j = 0; j < count - (i + 1); j++)
            rear = rear->next;
        if (front->data != rear->data)
            return false;
        else
            i++;
    }
    return true;
}

void getData(struct node **head_ref, char new_data)
{
    struct node *new_node = (struct node *)malloc(sizeof(struct node
));
    new_node->data = new_data;
    new_node->next = (*head_ref);
    (*head_ref) = new_node;
}

int main()
{
    struct node *head = NULL;
    char str[] = "Adarsh Kumar";
    int count = (sizeof(str) / sizeof(str[0])) - 1;

    for (int i = 0; str[i] != '\0'; i++)
    {
        getData(&head, str[i]);
    }
    isPalindrome(head, count) ? printf("Is Palindrome\n") : printf("
Not Palindrome\n");

    return 0;
}
```

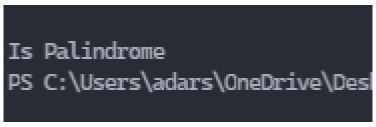
OUTPUT:-

Output for Adarsh Kumar is:-



```
PS C:\Users\adars\OneDrive\Desktop\3 rd sem> cd "c:\Use
Not Palindrome
PS C:\Users\adars\OneDrive\Desktop\3 rd sem\DSA Lab\DSA
```

Output for kanak is:-

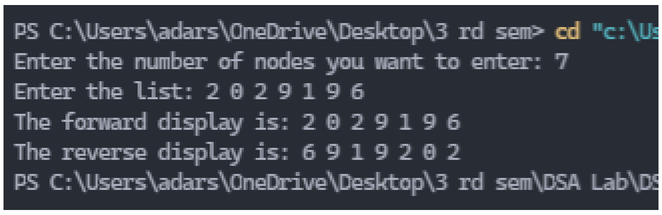


```
Is Palindrome
PS C:\Users\adars\OneDrive\Des
```

5. WAP to display the contents of a linked list in reverse order.

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

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

struct node *head = NULL, *tail = NULL;

void getData(struct node **current, int n)
{
    printf("Enter the list: ");
    for (int i = 0; i < n; i++)
    {
        *current = malloc(sizeof(struct node));
        scanf("%d", &(*current)->data);
        (*current)->next = NULL;
        if (head == NULL)
        {
            head = tail = *current;
        }
        else
        {
            tail->next = *current;
            tail = *current;
        }
    }
}

void reverseList(struct node **current)
{

    struct node *previous = NULL;
    struct node *next = NULL;
    *current = head;

    while ((*current) != NULL)
    {
        next = (*current)->next;
        (*current)->next = previous;
        previous = (*current);
        (*current) = next;
    }
    head = previous;
}

void displayList(struct node **current)
{
    *current = head;
    while (*current != NULL)
    {
        printf("%d ", (*current)->data);
        *current = (*current)->next;
    }
    printf("\n");
}

int main()
{
    int num;
    printf("Enter the number of nodes you want to enter: ");
    scanf("%d", &num);

    struct node *c;

    getData(&c, num);
    printf("The forward display is: ");
    displayList(&c);
    printf("The reverse display is: ");
    reverseList(&c);
    displayList(&c);

    return 0;
}
```

OUTPUT:-



```
PS C:\Users\adars\OneDrive\Desktop\3 rd sem> cd "c:\Us
Enter the number of nodes you want to enter: 7
Enter the list: 2 0 2 9 1 9 6
The forward display is: 2 0 2 9 1 9 6
The reverse display is: 6 9 1 9 2 0 2
PS C:\Users\adars\OneDrive\Desktop\3 rd sem\DSA Lab\DS
```

6. Given a singly linked list, rotate the linked list counter-clockwise by k nodes. Where k is a given positive integer. For example, if the given linked list is 10->20->30->40->50->60 and k is 4, the list should be modified to 50->60->10->20->30->40. Assume that k is smaller than the count of nodes in linked list.

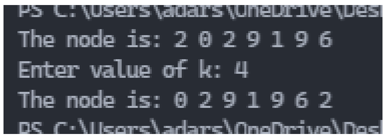
Roll no :- 2029196

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

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

struct node *head = NULL;

void create (int newData)
{
    struct node *newNode = malloc (sizeof (struct node));
    struct node * last = head;

    ~~newNode = newData~~
     newNode->data = newData;
     newNode ->next = NULL;

     if (head == NULL)
     {
         head = newNode;
             return;
     }

     while (last -> next != NULL)
             last = last->next;

      last -> next = newNode;
      return;
  }
  void   modified (int K)
  {
     struct node * current = head, *temp = current;

      if (K==0)
          return;
      int count =1;
      while (count < K && current != NULL ~~current != NULL~~)
      {
          current = current -> next;
          count ++;
      }
```

Roll no:- 2029196

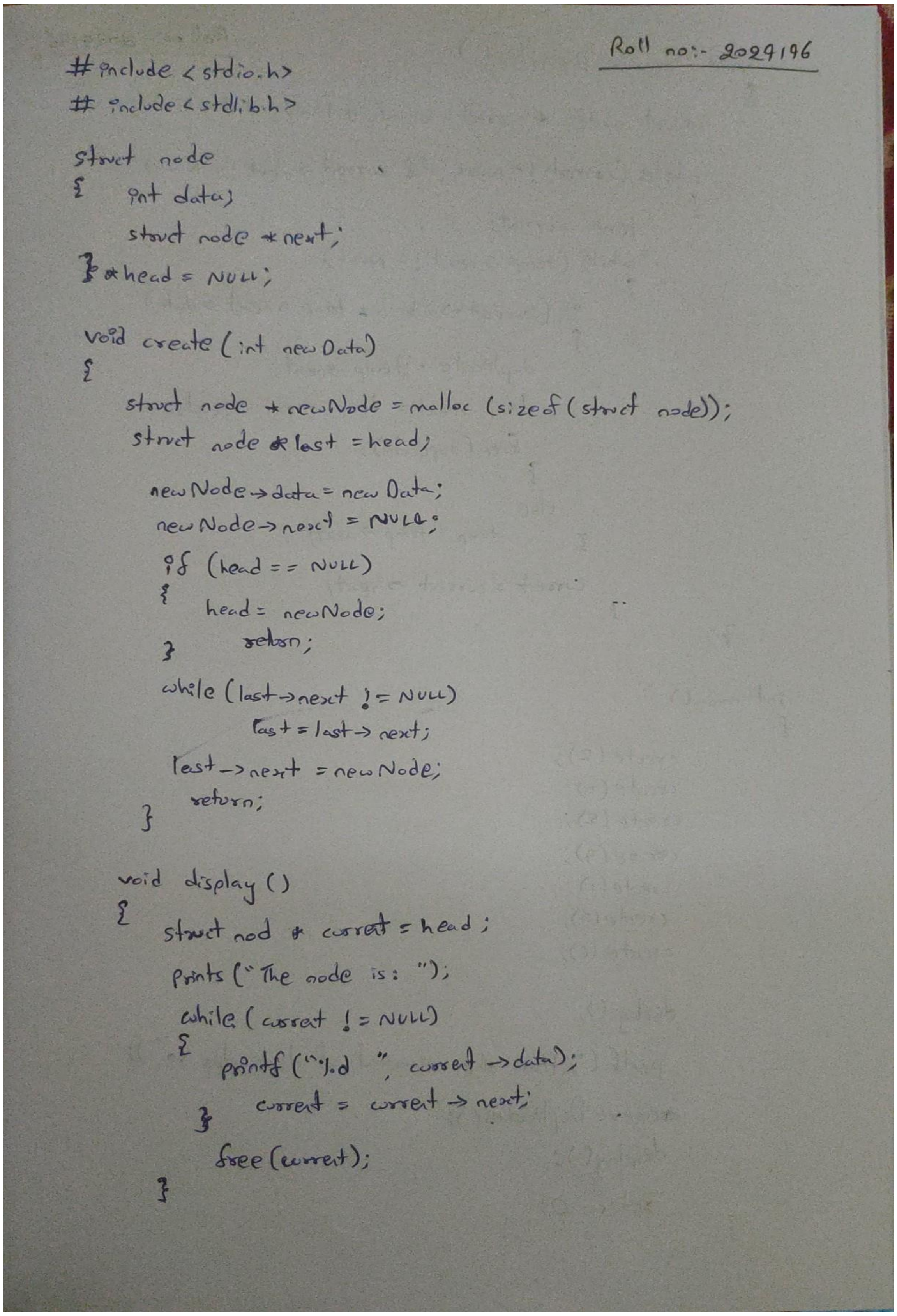
```
    if (count == NULL)
        return;
    while (currnt->next != NULL)
        current = currnt->next;
    currnt->next = head;
    head = temp->next;
    temp->next = NULL;
    display();
  }

int main()
{
        create(2);
        create(0);
        create(2);
        create(9);
        create(1);
        create(9);
        create(6);

        display.list();
        int n;
         Printf ("Enter the value of K: ");
         Scanf ("%d", &n);

         modified (n);

         return 0;
  }
```

OUTPUT:-



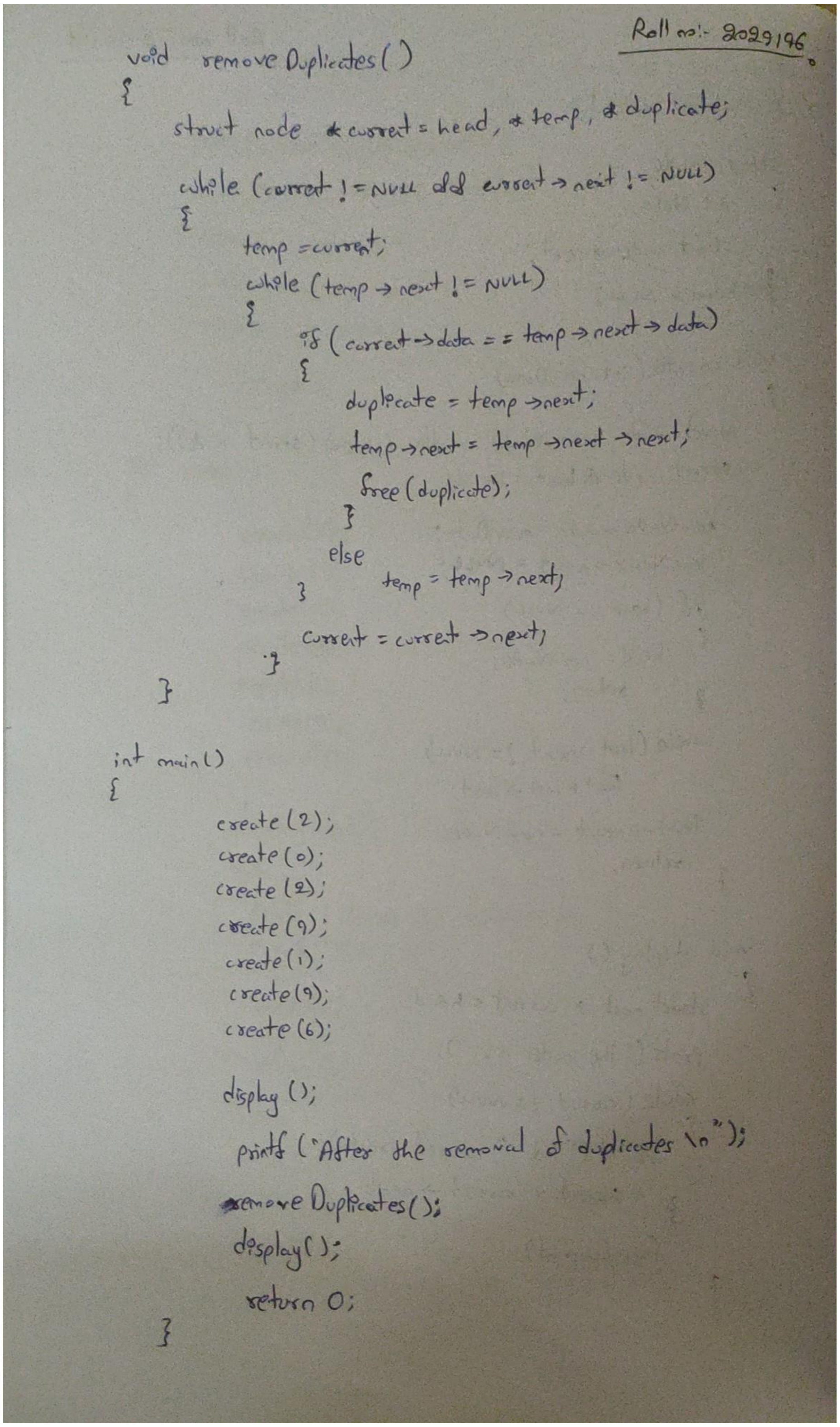
```
PS C:\Users\adars\OneDrive\Des
The node is: 2 0 2 9 1 9 6
Enter value of k: 4
The node is: 0 2 9 1 9 6 2
PS C:\Users\adars\OneDrive\Des
```

7. WAP to remove duplicates from a linked list of n nodes.



Roll no:- 2029196

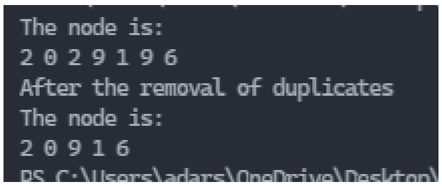
```
#include <stdio.h>
# include <stdlib.h>

struct node
{   int data;
    struct node *next;
} *head = NULL;

void create (int newData)
{
    struct node *newNode = malloc (sizeof (struct node));
    struct node *last = head;

     newNode->data = newData;
      newNode->next = NULL;
      if (head == NULL)
      {
          head = newNode;
             return;
      }
      while (last->next != NULL)
               last = last->next;
       last->next = newNode;
        return;
}

void display ()
{
    struct nod * currat = head;
     Prints ("The node is: ");
     while (currat != NULL)
     {
        printf ("%d ", curret->data);
          current = curret -> next;
     }
        free (current);
}
```



Roll no:- 2029196

```
void remove Duplicates( )
{
    struct node *current = head, *temp, *duplicate;

    while (current != NULL && current->next != NULL)
    {
        temp = current;
        while (temp->next != NULL)
        {
            if (current->data == temp->next->data)
            {
                duplicate = temp->next;
                temp->next = temp->next->next;
                free(duplicate);
            }
            else
                temp = temp->next;
        }
        current = current->next;
    }
}

int main()
{
        create(2);
        create(0);
        create(2);
        create(9);
        create(1);
        create(9);
        create(6);

        display();
        printf("After the removal of duplicates \n");
        remove Duplicates();
        display();
        return 0;
}
```

OUTPUT:-



```
The node is:
2 0 2 9 1 9 6
After the removal of duplicates
The node is:
2 0 9 1 6
PS C:\Users\adars\OneDrive\Desktop\
```