

ADVANCED C PROGRAMMING – MODULE 2

1. Write a C program to remove duplicate element from sorted Linked List.

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

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

struct node *head, *temp, *newnode;

void create(){
    int n;
    printf("Number of nodes: ");
    scanf("%d", &n);
    for(int i=0;i<n;i++){
        newnode = (struct node *)malloc(sizeof(struct node));
        printf("Data = ");
        scanf("%d", &newnode->data);
        newnode->next = NULL;
        if(head==NULL){
            head=temp=newnode;
        }
        else{
            temp->next=newnode;
            temp=newnode;
        }
    }
}

void display(){
    temp=head;
    while(temp!=NULL){
        printf("%d -> ", temp->data);
        temp= temp->next;
    }
    printf("NULL");
}

void rmdup(){
    struct node* temp1;
    temp=head;
    while(temp->next!=NULL){
        if(temp->data==temp->next->data){
            temp1=temp->next->next;
            free(temp->next);
            temp->next=temp1;
        }
    }
}
```

```

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

int main(){
    printf("Linked List Creation\n");
    create();
    printf("Linked list: ");
    display();
    rmdup();
    printf("\nlinked list after removing duplicates: ");
    display();
}

```

```

main.c
32 while(temp!=NULL){
33     printf("%d -> ", temp->data);
34     temp= temp->next;
35 }
36 printf("NULL");
37 }
38
39 void rmdup(){
40     struct node* temp1;
41     temp= head;
42     while(temp->next!=NULL){
43         if(temp->data==temp->next->data){
44             temp1= temp->next->next;
45             free(temp->next);
46             temp->next=temp1;
47         }
48         else{
49             temp=temp->next;
50         }
51     }
52 }
53 int main(){
54     printf("Linked List Creation\n");
55     create();
56     printf("Linked list: ");
57     display();
58     rmdup();
59     printf("\nlinked list after removing duplicates: ");
60     display();
61 }

```

```

input
Linked List Creation
Number of nodes: 4
Data = 2
Data = 3
Data = 3
Data = 4
Linked list: 2 -> 3 -> 3 -> 4 -> NULL
linked list after removing duplicates: 2 -> 3 -> 4 -> NULL
...Program finished with exit code 0
Press ENTER to exit console.

```

2. Write a C program to rotate a doubly linked list by N nodes.

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

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

```

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

```

```
struct node *head, *temp, *newnode;
```

```

void create(){
    int n;
    printf("Enter the number of nodes: ");
    scanf("%d", &n);
}

```

```

for(int i=0;i<n;i++){
    newnode = (struct node *)malloc(sizeof(struct node));
    printf("Data = ");
    scanf(" %c", &newnode->data);
    newnode->prev = NULL;
    newnode->next = NULL;
    if(head==NULL){
        head=temp=newnode;
    }
    else{
        temp->next=newnode;
        newnode->prev=temp;
        temp=newnode;
    }
}
}

```

```

void display(){
    temp=head;
    while(temp!=NULL){
        printf("%c-> ", temp->data);
        temp= temp->next;
    }
    printf("NULL\n");
}

```

```

void rotate(){
    int n;
    printf("Number of positions to be rotated = ");
    scanf("%d", &n);
    if(n==0){
        return;
    }
    temp=head;
    while(temp->next!=NULL){
        temp=temp->next;
    }
    temp->next=head;
    head->prev=temp;
    int count=1;
    while(count<=n){
        head=head->next;
        temp=temp->next;
        count++;
    }
    temp->next=NULL;
    head->prev=NULL;
    printf("\nDoubly linked list after rotation: ");
    display();
}

```

```

}
int main(){
    create();
    printf("Doubly linked list: ");
    display();
    rotate();
}

```

```

main.c
42 void rotate(){
43     int n;
44     printf("Number of positions to be rotated = ");
45     scanf("%d", &n);
46     if(n==0){
47         return;
48     }
49     temp= head;
50     while(temp->next!=NULL){
51         temp=temp->next;
52     }
53     temp->next=head;
54     head->prev=temp;
55     int count=1;
56     while(count<=n){
57         head=head->next;
58         temp=temp->next;
59         count++;
60     }
61     temp->next=NULL;
62     head->prev=NULL;
63     printf("\nDoubly linked list after rotation: ");
64     display();
65 }

```

input

```

Enter the number of nodes: 5
Data = a
Data = b
Data = c
Data = d
Data = e
Doubly linked list: a-> b-> c-> d-> e-> NULL
Number of positions to be rotated = 2

Doubly linked list after rotation:  c-> d-> e-> a-> b-> NULL

...Program finished with exit code 0
Press ENTER to exit console.

```

3. Write a C program to sort the elements of a queue in ascending order.

```

#include <stdio.h>

#define N 5

int queue[N];
int front = -1;
int rear = -1;

void enqueue(int x){
    if(rear==N-1){
        printf("overflow\n");
    }
    else if((front==-1)&&(rear==-1)){
        front=rear=0;
        queue[rear]=x;
    }
    else{
        rear++;
        queue[rear]=x;
    }
}

void dequeue(){
    if((front==-1)&&(rear==-1)){

```

```

        printf("\nUnderflow");
    }
    else if(front==rear){
        front=rear=-1;
    }
    else{
        front++;
    }
}

void display(){
    if((front==-1)&&(rear==-1)){
        printf("\nQueue is empty");
    }
    else{
        for(int i=front;i<rear+1;i++){
            printf("%d ", queue[i]);
        }
    }
}

void sort(){
    int n=rear - front +1;
    int i, j, temp;
    for(i=0;i<n-1;i++){
        for(j=i+1;j<n;j++){
            if(queue[i]>queue[j]){
                temp=queue[i];
                queue[i]=queue[j];
                queue[j]=temp;
            }
        }
    }
}

void main(){
    enqueue(4);
    enqueue(2);
    enqueue(7);
    enqueue(5);
    enqueue(1);
    printf("\nBefore sorting: ");
    display();
    sort();
    printf("\nAfter sorting: ");
    display();
}

```

```

main.c
41 }
42 }
43
44 void sort(){
45     int n=rear - front +1;
46     int i, j, temp;
47     for(i=0;i<n-1;i++){
48         for(j=i+1;j<n;j++){
49             if(queue[i]>queue[j]){
50                 temp=queue[i];
51                 queue[i]=queue[j];
52                 queue[j]=temp;
53             }
54         }
55     }
56 }
57 void main(){
58     enqueue(4);
59     enqueue(2);
60     enqueue(7);
61     enqueue(5);
62     enqueue(1);
63     printf("\nBefore sorting: ");
64     display();
}

input
Before sorting: 4 2 7 5 1
After sorting: 1 2 4 5 7
...Program finished with exit code 4
Press ENTER to exit console.

```

4. List all queue function operations available for manipulation of data elements in C

- enqueue() – Add an element to the end of the queue
- dequeue() – Remove an element from the front of the queue
- display() – Display all elements in a queue
- peek() – Get the value of the front of the queue without removing it
- isfull() – Checks if the queue is full
- isempty() – Checks if the queue is empty

5. Reverse the given string using stack

```

#include <stdio.h>
#define N 100
char stack[N];
int top=-1;
void push(char x){
    if(top==N-1){
        printf("Overflow");
    }
    else{
        top++;
        stack[top]=x;
    }
}
void pop(){
    if(top==-1){
        printf("Underflow");
    }
    else{
        printf("%c", stack[top]);
        top--;
    }
}
}

```

```

int main(){
    char string[N];
    printf("Enter the string to be reversed: ");
    scanf("%s", string);
    int size = sizeof(string)/sizeof(string[0]);
    for(int i=0;i<size;i++){
        push(string[i]);
    }
    printf("The reversed string: ");
    for(int i=0;i<size;i++){
        pop();
    }
}

```

```

main.c
3 char stack[N];
4 int top=-1;
5 void push(char x){
6     if(top==N-1){
7         printf("Overflow");
8     }
9     else{
10        top++;
11        stack[top]=x;
12    }
13 }
14 void pop(){
15     if(top==-1){
16         printf("Underflow");
17     }
18     else{
19         printf("%c", stack[top]);
20         top--;
21     }
22 }
23 int main(){
24     char string[N];
25     printf("Enter the string to be reversed: ");
26     scanf("%s", string);
27     int size = sizeof(string)/sizeof(string[0]);
28     for(int i=0;i<size;i++){
29         push(string[i]);
30     }
31     printf("The reversed string: ");
32     for(int i=0;i<size;i++){
33         pop();
34     }
35 }

```

input

Enter the string to be reversed: Letslearn
The reversed string: nraelstEL
...Program finished with exit code 0
Press ENTER to exit console.

6. Insert value in sorted way in a sorted doubly linked list. Given a sorted doubly linked list and a value to insert, write a function to insert the value in sorted way.

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

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

```

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

```

```
struct node *head, *temp, *newnode;
```

```

void create(){
    int n, i;
    printf("Number of nodes = ");
    scanf("%d", &n);
}

```

```

printf("Enter the elements in a sorted way\n");
for(i=0;i<n;i++){
    newnode = (struct node *)malloc(sizeof(struct node));
    printf("Data = ");
    scanf("%d", &newnode->data);
    newnode->prev = NULL;
    newnode->next = NULL;
    if(head==NULL){
        head=temp=newnode;
    }
    else{
        temp->next=newnode;
        newnode->prev=temp;
        temp=newnode;
    }
}
}

void insert_at_pos(){
    int x;
    printf("Enter the element to be inserted: ");
    scanf("%d", &x);
    struct node* new_insert = (struct node *)malloc(sizeof(struct node));
    new_insert->data=x;
    new_insert->prev=NULL;
    new_insert->next=NULL;
    temp=head;
    while(temp != NULL) {
        if(x < temp->data) {
            new_insert->next = temp;
            if(temp->prev != NULL) {
                temp->prev->next = new_insert;
                new_insert->prev = temp->prev;
            } else {
                head = new_insert;
            }
            temp->prev = new_insert;
            break;
        } else if(temp->next == NULL) {
            temp->next = new_insert;
            new_insert->prev = temp;
            break;
        }
        temp = temp->next;
    }
}

```

```

void display(){
    temp=head;

```



```

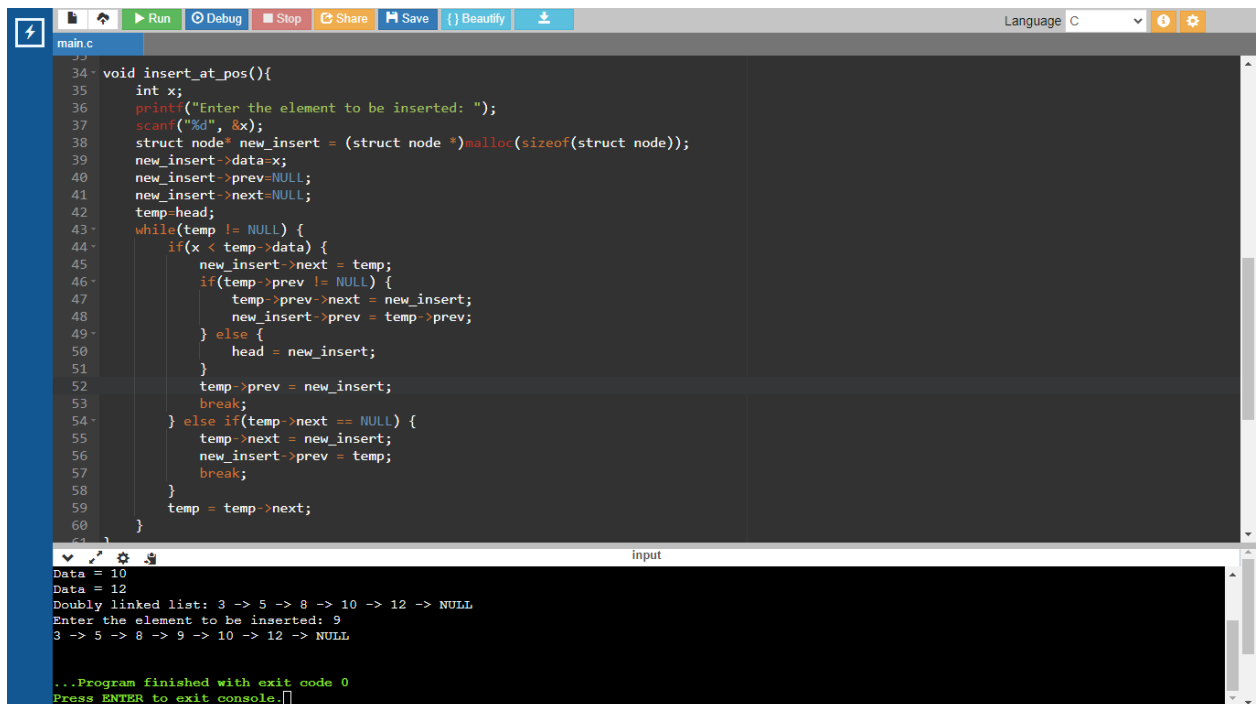
while(temp!=NULL){
    printf("%d -> ", temp->data);
    temp= temp->next;
}
printf("NULL\n");
}

```

```

int main(){
    create();
    printf("Doubly linked list: ");
    display();
    insert_at_pos();
    display();
}

```



```

main.c
34 void insert_at_pos(){
35     int x;
36     printf("Enter the element to be inserted: ");
37     scanf("%d", &x);
38     struct node* new_insert = (struct node *)malloc(sizeof(struct node));
39     new_insert->data=x;
40     new_insert->prev=NULL;
41     new_insert->next=NULL;
42     temp=head;
43     while(temp != NULL) {
44         if(x < temp->data) {
45             new_insert->next = temp;
46             if(temp->prev != NULL) {
47                 temp->prev->next = new_insert;
48                 new_insert->prev = temp->prev;
49             } else {
50                 head = new_insert;
51             }
52             temp->prev = new_insert;
53             break;
54         } else if(temp->next == NULL) {
55             temp->next = new_insert;
56             new_insert->prev = temp;
57             break;
58         }
59         temp = temp->next;
60     }
}

input
Data = 10
Data = 12
Doubly linked list: 3 -> 5 -> 8 -> 10 -> 12 -> NULL
Enter the element to be inserted: 9
3 -> 5 -> 8 -> 9 -> 10 -> 12 -> NULL

...Program finished with exit code 0
Press ENTER to exit console.

```

Output: 8

7. Write a C program to insert/delete and count the number of elements in a queue.

```

#include <stdio.h>

#define N 5

int queue[N];
int front = -1;
int rear = -1;

void enqueue(int x){
    if(rear==N-1){
        printf("Overflow\n");
    }
    else if((front==-1)&&(rear==-1)){
        front=rear=0;
        queue[rear]=x;
    }
    else{

```

```

        rear++;
        queue[rear]=x;
    }
}

void dequeue(){
    if((front== -1)&&(rear== -1)){
        printf("\nUnderflow");
    }
    else if(front==rear){
        front=rear=-1;
    }
    else{
        front++;
    }
}

void display(){
    if((front== -1)&&(rear== -1)){
        printf("\nQueue is empty");
    }
    else{
        for(int i=front;i<rear+1;i++){
            printf("%d ", queue[i]);
        }
        printf("\n");
    }
}

void count(){
    int countVar=0;
    if((front== -1)&&(rear== -1)){
        countVar=0;
    }
    else{
        for(int i=front;i<=rear;i++){
            countVar++;
        }
    }
    printf("Number of elements in the queue: %d\n", countVar);
}

void main(){
    printf("Initialize a queue!\n");
    count();
    printf("Insert some elements into the queue: \n");
    enqueue(1);
    enqueue(2);
    enqueue(3);
    printf("The queue elements are: \n");
}

```

```

display();
printf("Delete two elements from the said queue: \n");
dequeue();
dequeue();
printf("The queue elements are: \n");
display();
count();
printf("Insert another element into the queue: \n");
enqueue(4);
printf("The queue elements are: \n");
display();
count();
}

```

```

main.c
7 void enqueue(int x){
8     if(rear==N-1){
9         printf("Overflow\n");
10    }
11    else if((front==1)&&(rear==1)){
12        front=rear=0;
13        queue[rear]=x;
14    }
15    else{
16        rear++;
17        queue[rear]=x;
18    }
19 }
20
21 void dequeue(){
22     if((front==1)&&(rear==1)){
23         printf("\nUnderFlow");
24     }
25     else if(front==rear){
26         front=rear--1;
27     }
28     else{
29         front++;
30     }
31 }
32
33 void display(){
34     if((front==1)&&(rear==1)){
35         printf("\nQueue is empty");
36     }
37     else{
38         for(int i=front;i<rear+1;i++){
39             printf("%d ", queue[i]);
40         }
41     }
42     printf("\n");
43 }

```

```

main.c
42     printf("\n");
43 }
44
45 void count(){
46     int countVar=0;
47     if((front==1)&&(rear==1)){
48         countVar=0;
49     }
50     else{
51         for(int i=front;i<=rear;i++){
52             countVar++;
53         }
54     }
55     printf("Number of elements in the queue: %d\n", countVar);
56 }
57 void main(){
58     printf("Initialize a queue!\n");
59     count();
60     printf("Insert some elements into the queue: \n");

```

```

input
Initialize a queue!
Number of elements in the queue: 0
Insert some elements into the queue:
The queue elements are:
1 2 3
Delete two elements from the said queue:
The queue elements are:
3
Number of elements in the queue: 1
Insert another element into the queue:
The queue elements are:
3 4
Number of elements in the queue: 2

...Program finished with exit code 35
Press ENTER to exit console.

```

8. Write a C program to Find whether an array is a subset of another array.

```
#include <stdio.h>
#include <stdbool.h>
#define MAX 100

bool isSubset(int arr1[], int m, int arr2[], int n) {
    bool hashset[MAX] = {false};
    for (int i = 0; i < m; i++) {
        hashset[arr1[i]] = true;
    }
    for (int i = 0; i < n; i++) {
        if (!hashset[arr2[i]])
            return false;
    }
    return true;
}

int main() {
    int m,n;
    printf("Enter the size of 1st array: ");
    scanf("%d", &m);
    int arr1[m];
    printf("Enter the elements of 1st array: ");
    for(int i=0;i<m;i++){
        scanf("%d", &arr1[i]);
    }

    printf("Enter the size of subset array: ");
    scanf("%d", &n);
    int arr2[n];
    printf("Enter the elements of subset array: ");
    for(int i=0;i<n;i++){
        scanf("%d", &arr2[i]);
    }

    if (isSubset(arr1, m, arr2, n))
        printf("arr2[] is subset of arr1[]\n");
    else
        printf("arr2[] is not a subset of arr1[]\n");

    return 0;
}
```

```
main.c
5 bool isSubset(int arr1[], int m, int arr2[], int n) {
6     bool hashset[MAX] = {false};
7     for (int i = 0; i < m; i++) {
8         hashset[arr1[i]] = true;
9     }
10    for (int i = 0; i < n; i++) {
11        if (!hashset[arr2[i]])
12            return false;
13    }
14    return true;
15 }
16 int main() {
17     int m,n;
18     printf("Enter the size of 1st array: ");
19     scanf("%d", &m);
20     int arr1[m];
21     printf("Enter the elements of 1st array: ");
22     for(int i=0;i<m;i++){
23         scanf("%d", &arr1[i]);
24     }
25
26     printf("Enter the size of subset array: ");
27     scanf("%d", &n);
28     int arr2[n];
29     printf("Enter the elements of subset array: ");
30     for(int i=0;i<n;i++){
31         scanf("%d", &arr2[i]);
32     }
33 }
34
```

input

```
Enter the size of 1st array: 6
Enter the elements of 1st array: 1 2 3 4 5 6
Enter the size of subset array: 4
Enter the elements of subset array: 3 4 5 2
arr2[] is subset of arr1[]

...Program finished with exit code 0
Press ENTER to exit console.
```

```
main.c
5 bool isSubset(int arr1[], int m, int arr2[], int n) {
6     bool hashset[MAX] = {false};
7     for (int i = 0; i < m; i++) {
8         hashset[arr1[i]] = true;
9     }
10    for (int i = 0; i < n; i++) {
11        if (!hashset[arr2[i]])
12            return false;
13    }
14    return true;
15 }
16 int main() {
17     int m,n;
18     printf("Enter the size of 1st array: ");
19     scanf("%d", &m);
20     int arr1[m];
21     printf("Enter the elements of 1st array: ");
22     for(int i=0;i<m;i++){
23         scanf("%d", &arr1[i]);
24     }
25
26     printf("Enter the size of subset array: ");
27     scanf("%d", &n);
28     int arr2[n];
29     printf("Enter the elements of subset array: ");
30     for(int i=0;i<n;i++){
31         scanf("%d", &arr2[i]);
32     }
33 }
34
```

input

```
Enter the size of 1st array: 3
Enter the elements of 1st array: 1 2 3
Enter the size of subset array: 2
Enter the elements of subset array: 5 6
arr2[] is not a subset of arr1[]

...Program finished with exit code 0
Press ENTER to exit console.
```