1. Write a program in C to create and display a Singly link list.

```
as3ds1.c > traversal(node *)
#include<stdio.h>
#include<stdlib.h>
struct node
{
    int data;
    struct node*next;
};
void traversal(struct node *ptr)
{   while (ptr!=NULL)
{
    printf("ELEMENT = %d\n",ptr->data);
    ptr=ptr->next;
}
};

int main(){
    //so i allocated my memory of linked list in heap
    struct node*head;
    head=( struct node*)malloc(sizeof( struct node));
    struct node*second;
    second=( struct node*)malloc(sizeof( struct node));
    struct node*third;
    third=( struct node*)malloc(sizeof( struct node));
struct node*fourth;
    fourth=( struct node*)malloc(sizeof( struct node));

head->data=7;
head->next=second;
second->data=8;
second->next=third;
third->data=9;
third->next=NULL;
traversal(head);
return 0;
}
```

2. Write a program in C to insert a new node at the beginning of a Singly Linked List.

as3ds2.c > insertbeg(node *, int)

```c
struct node
{
    int data;
    struct node *next;
};
void traverse(struct node *ptr)
{
    while (ptr != NULL)
    {
        printf("ELEMENT = %d\n", ptr->data);
        ptr = ptr->next;
    }
};
struct node *insertbeg(struct node *head, int data)
{
    struct node *ptr = (struct node *)malloc(sizeof(struct node));
    ptr->data = data;
    ptr->next = head;
    return ptr;
}
int main()
{
    struct node *head;
    head = (struct node *)malloc(sizeof(struct node *));
    struct node *second;
    second = (struct node *)malloc(sizeof(struct node *));
    struct node *third;
    third = (struct node *)malloc(sizeof(struct node *));
    struct node *fourth;
    fourth = (struct node *)malloc(sizeof(struct node *));
    head->data = 7;
    head->next = second;
    second->data = 8;
    second->next = third;
    third->data = 9;
    third->next = fourth;
    fourth->data = 10;
    fourth->next = NULL;
    traverse(head);
    printf("\n");
    head = insertbeg(head, 6);
    traverse(head);
    return 0;
}
```

3. Write a program in C to traverse in a singly linked list.

as3ds3.c > main()

```c
#include<stdio.h>
#include<stdlib.h>
struct node
{
    int data;
    struct node*next;
};
void traversal(struct node *ptr)
{   while (ptr!=NULL)
{
    printf("ELEMENT = %d\n",ptr->data);
    ptr=ptr->next;
}
};

int main(){
    //so i allocated my memory of linked list in heap
    struct node*head;
    head=( struct node*)malloc(sizeof( struct node));
    struct node*second;
    second=( struct node*)malloc(sizeof( struct node));
    struct node*third;
    third=( struct node*)malloc(sizeof( struct node));
struct node*fourth;
    fourth=( struct node*)malloc(sizeof( struct node));

head->data=7;
head->next=second;
second->data=8;
second->next=third;
third->data=9;
third->next=NULL;
traversal(head);
return 0;
}
```

4. Write a program in C to copy the elements of the array to a singly linked list.

as3ds4.c > main()

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

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


struct Node* createNode(int data) {
    struct Node* newNode = (struct Node*)malloc(sizeof(struct Node));
    newNode->data = data;
    newNode->next = NULL;
    return newNode;
}

void appendNode(struct Node** head, int data) {
    struct Node* newNode = createNode(data);
    if (*head == NULL) {
        *head = newNode;
    } else {
        struct Node* temp = *head;
        while (temp->next != NULL) {
            temp = temp->next;
        }
        temp->next = newNode;
    }
}


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


void arrayToLinkedList(int arr[], int size, struct Node** head) {
    for (int i = 0; i < size; i++) {
        appendNode(head, arr[i]);
    }
}

int main() {
    int arr[] = {1, 2, 3, 4, 5};
    int size = sizeof(arr) / sizeof(arr[0]);

    struct Node* head = NULL;


    arrayToLinkedList(arr, size, &head);


    printLinkedList(head);

    return 0;
}
```

5. Write a C program that converts a singly linked list into an array and returns it.

```c
struct Node* createNode(int data) {
    struct Node* newNode = (struct Node*)malloc(sizeof(struct Node));
    newNode->data = data;
    newNode->next = NULL;
    return newNode;
}


void appendNode(struct Node** head, int data) {
    struct Node* newNode = createNode(data);
    if (*head == NULL) {
        *head = newNode;
    } else {
        struct Node* temp = *head;
        while (temp->next != NULL) {
            temp = temp->next;
        }
        temp->next = newNode;
    }
}


int* linkedListToArray(struct Node* head, int* size) {
    int count = 0;
    struct Node* temp = head;
    while (temp != NULL) {
        count++;
        temp = temp->next;
    }

    int* arr = (int*)malloc(count * sizeof(int));
    temp = head;
    for (int i = 0; i < count; i++) {
        arr[i] = temp->data;
        temp = temp->next;
    }

    *size = count;
    return arr;
}


void printArray(int arr[], int size) {
    for (int i = 0; i < size; i++) {
        printf("%d ", arr[i]);
    }
    printf("\n");
}

int main() {
    struct Node* head = NULL;

    appendNode(&head, 1);
    appendNode(&head, 2);
    appendNode(&head, 3);
    appendNode(&head, 4);
    appendNode(&head, 5);

    int size;
    int* arr = linkedListToArray(head, &size);

    printArray(arr, size);


    free(arr);
    return 0;
}
```