EXPERIMENT10: DYNAMIC MEMORY ALLOCATION

Experiment 1: Creation of a Simple Linked List

Aim

To create a simple singly linked list using dynamic memory allocation with pointers and structures.

Algorithm

1. Define a node structure containing data and next pointer
2. Use `malloc()` to create nodes dynamically
3. Link nodes using pointers
4. Display the linked list

Pseudocode

```
STRUCT Node
    data
    next
END STRUCT

CREATE nodes using malloc
```

```
LINK nodes
DISPLAY list
```

Flowchart (ASCII)

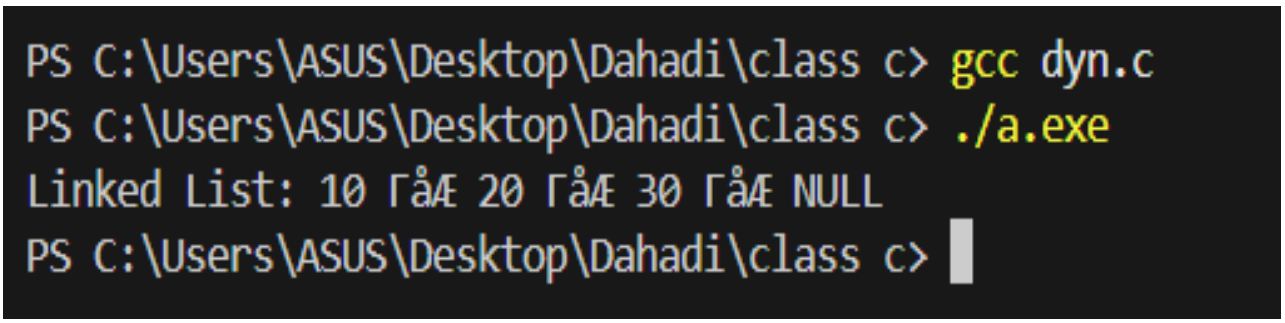
```
Start
   ↓
Create First Node
   ↓
Create Next Node
   ↓
Link Nodes
   ↓
Display List
   ↓
 End
```

C Program

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

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

int main() {
    struct Node *head, *second, *third;

    head = (struct Node *)malloc(sizeof(struct Node));
    second = (struct Node *)malloc(sizeof(struct Node));
    third = (struct Node *)malloc(sizeof(struct Node));
```

```
    head->data = 10;
    head->next = second;

    second->data = 20;
    second->next = third;

    third->data = 30;
    third->next = NULL;

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

    return 0;
}
```

Output



Conclusion

A singly linked list was successfully created using dynamic memory allocation and pointers.

Experiment 2: Insertion in the Middle of a Linked List

Aim

To insert a new element at the middle position of a singly linked list.

Algorithm

1. Create initial linked list
2. Find middle position
3. Create a new node using `malloc()`
4. Insert the new node by adjusting pointers
5. Display updated list

Pseudocode

```
CREATE linked list
CALCULATE length
mid = length / 2
MOVE pointer to mid-1
```

```
INSERT new node
DISPLAY list
```

Flowchart (ASCII)

```
Start
   ↓
Create Linked List
   ↓
Find Middle
   ↓
Create New Node
   ↓
Insert Node
   ↓
Display List
   ↓
 End
```

C Program

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

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

void display(struct Node *head) {
    while (head != NULL) {
        printf("%d → ", head->data);
        head = head->next;
```

```
    }
    printf("NULL\n");
}

int main() {
    struct Node *head, *second, *third, *newNode;
    head = (struct Node *)malloc(sizeof(struct Node));
    second = (struct Node *)malloc(sizeof(struct Node));
    third = (struct Node *)malloc(sizeof(struct Node));

    head->data = 10;
    head->next = second;

    second->data = 20;
    second->next = third;

    third->data = 30;
    third->next = NULL;

    printf("Original List: ");
    display(head);

    newNode = (struct Node *)malloc(sizeof(struct Node));
    newNode->data = 25;

    newNode->next = second->next;
    second->next = newNode;

    printf("After Insertion in Middle: ");
    display(head);

    return 0;
}
```

Output

```
PS C:\Users\ASUS\Desktop\Dahadi\class c> gcc dyn.c
PS C:\Users\ASUS\Desktop\Dahadi\class c> ./a.exe
Original List: 10 ΓåÆ 20 ΓåÆ 30 ΓåÆ NULL
After Insertion in Middle: 10 ΓåÆ 20 ΓåÆ 25 ΓåÆ 30 ΓåÆ NULL
PS C:\Users\ASUS\Desktop\Dahadi\class c>
```

Conclusion

Insertion in the middle of a linked list was successfully implemented using dynamic memory allocation and pointer manipulation.