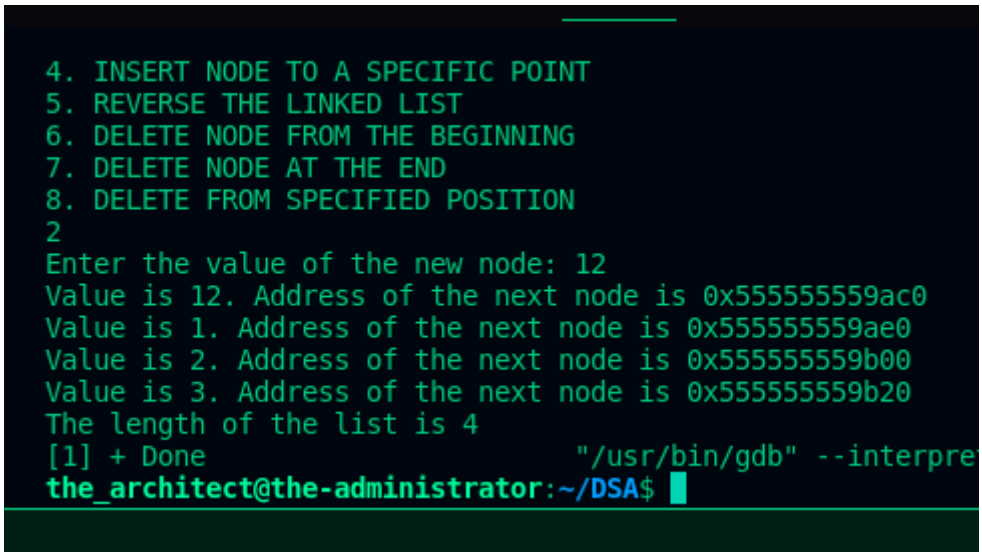
22AIE112 Data Structure And Algorithms
Labsheet 4
Circular Linked List

Date: 3/7/2023 **RollNo: AM.EN.U4AIE22009**

1.Instert at beginnig

```c
void insertbegin() {
    struct node* newnode;
    int data;

    newnode = (struct node*)malloc(sizeof(struct node));
    if (newnode == NULL) {
        printf("Unable to allocate memory.");
        return;
    }

    printf("Enter the value of the new node: ");
    scanf("%d", &data);

    newnode->data = data;
    newnode->link = head;

    // Update the last node to point to the new node
    struct node* last = head;
    while (last->link != head) {
        last = last->link;
    }
    last->link = newnode;

    head = newnode;

    traversal();
}
```



2.Insert at end

```c
// Function to insert a node at the end
void insertend() {
    struct node* newnode, *temp;
    int data;

    newnode = (struct node*)malloc(sizeof(struct node));
    if (newnode == NULL) {
        printf("Unable to allocate memory.");
        return;
    }

    printf("Enter the value of the new node: ");
    scanf("%d", &data);

    newnode->data = data;
    newnode->link = head;

    if (head == NULL) {
        head = newnode;
        newnode->link = head;
    } else {
        temp = head;
        while (temp->link != head) {
            temp = temp->link;
        }
        temp->link = newnode;
    }

    traversal();
}
```

```
4. INSERT NODE TO A SPECIFIC POINT
5. REVERSE THE LINKED LIST
6. DELETE NODE FROM THE BEGINNING
7. DELETE NODE AT THE END
8. DELETE FROM SPECIFIED POSITION
3
Enter the value of the new node: 12
Value is 1. Address of the next node is 0x555555559ae0
Value is 2. Address of the next node is 0x555555559b00
Value is 3. Address of the next node is 0x555555559b20
Value is 12. Address of the next node is 0x555555559ac0
The length of the list is 4
[1] + Done                       "/usr/bin/gdb" --interprete
the_architect@the-administrator:~/DSA$
```

3. Insert at position

```c
void insertpos() {
    int position, counter = 1;
    int data;
    struct node* newnode, *temp, *prev;

    newnode = (struct node*)malloc(sizeof(struct node));
    if (newnode == NULL) {
        printf("Unable to allocate memory.");
        return;
    }

    printf("Enter the position where to insert: ");
    scanf("%d", &position);

    printf("Enter the value of the new node: ");
    scanf("%d", &data);

    newnode->data = data;
    newnode->link = NULL;

    temp = head->link;
    prev = temp;


    while (temp != head && counter < position) {
        prev = temp;
        temp = temp->link;
        counter++;
    }

    if (temp == head) {
        printf("Invalid position. Inserting at the end.\n");
        prev->link = newnode;
        newnode->link = head;
    } else {
        newnode->link = temp;
        if (prev == NULL) {
            struct node* last = head;
            while (last->link != head) {
                last = last->link;
            }
            last->link = newnode;
            head = newnode;
        } else {
            prev->link = newnode;
        }
    }

    traversal();
}
```

```
8. DELETE FROM SPECIFIED POSITION
4
Enter the position where to insert: 2
Enter the value of the new node: 12
Value is 1. Address of the next node is 0x555555559ae0
Value is 2. Address of the next node is 0x555555559b20
Value is 12. Address of the next node is 0x555555559b00
Value is 3. Address of the next node is 0x555555559ac0
The length of the list is 4
```

4.Delete at beginnig

```c
void deletebeg() {
    if (head == NULL) {
        printf("List is empty. Unable to delete.\n");
        return;
    }

    struct node* temp = head;
    struct node* last = head;

    while (last->link != head) {
        last = last->link;
    }

    if (head == last) {
        head = NULL;
    } else {
        head = head->link;
        last->link = head;
    }

    free(temp);

    traversal();
}
```

```
5. REVERSE THE LINKED LIST
6. DELETE NODE FROM THE BEGINNING
7. DELETE NODE AT THE END
8. DELETE FROM SPECIFIED POSITION
6
Value is 2. Address of the next node is 0x555555559b00
Value is 3. Address of the next node is 0x555555559ae0
The length of the list is 2
[1] + Done                       "/usr/bin/gdb" --interpret
the_architect@the-administrator:~/DSA$
```

5. Delete at end

```c
// Function to delete the last node
void deleteend() {
    struct node* temp = head;
    struct node* prev = NULL;

    if (temp == NULL) {
        printf("List is empty. Unable to delete.\n");
        return;
    }

    while (temp->link != head) {
        prev = temp;
        temp = temp->link;
    }

    if (prev == NULL) {
        head = NULL;
    } else {
        prev->link = head;
    }

    free(temp);

    traversal();
}
```

```
6. DELETE NODE FROM THE BEGINNING
7. DELETE NODE AT THE END
8. DELETE FROM SPECIFIED POSITION
7
Value is 1. Address of the next node is 0x555555559ae0
Value is 2. Address of the next node is 0x555555559ac0
The length of the list is 2
[1] + Done                       "/usr/bin/gdb" --interpreter=mi
the_architect@the-administrator:~/DSA$
```

6. Delete at position

```c
// Function to delete a node at a specific positi
void deletepos() {
    int position, counter = 1;
    struct node* temp = head;
    struct node* prev = NULL;

    if (temp == NULL) {
        printf("List is empty. Unable to delete.\r
        return;
    }

    printf("Enter the position to delete: ");
    scanf("%d", &position);

    while (temp->link != head && counter < positi
        prev = temp;
        temp = temp->link;
        counter++;
    }

    if (temp == head) {
        printf("Invalid position. Unable to delete
        return;
    }

    if (prev == NULL) {
        struct node* last = head;
        while (last->link != head) {
            last = last->link;
        }
        head = head->link;
        last->link = head;
    } else {
        prev->link = temp->link;
    }

    free(temp);

    traversal();
}
```

```
8. DELETE FROM SPECIFIED POSITION
8
Enter the position to delete: 2
Value is 1. Address of the next node is 0x555555559b00
Value is 3. Address of the next node is 0x555555559ac0
The length of the list is 2
[1] + Done                       "/usr/bin/gdb" --interprete
the_architect@the-administrator:~/DSA$
```

7. Reverse the Circular Linked List

```c
// Function to reverse the circular linked list
void reverse() {
    struct node* prev = NULL;
    struct node* current = head;
    struct node* next = NULL;

    do {
        next = current->link;
        current->link = prev;
        prev = current;
        current = next;
    } while (current != head);

    head->link = prev;
    head = prev;

    traversal();
}
```

```
Enter the number of nodes: 5
Enter the data of node 1: 1
Enter the data of node 2: 2
Enter the data of node 3: 3
Enter the data of node 4: 4
Enter the data of node 5: 5
Circular Linked List OPERATION MENU
1. FIND LENGTH OF LINKED LIST
2. INSERT NODE TO BEGINNING
3. INSERT NODE TO THE END
4. INSERT NODE TO A SPECIFIC POINT
5. REVERSE THE LINKED LIST
6. DELETE NODE FROM THE BEGINNING
7. DELETE NODE AT THE END
8. DELETE FROM SPECIFIED POSITION
5
Value is 5. Address of the next node is 0x555555559b20
Value is 4. Address of the next node is 0x555555559b00
Value is 3. Address of the next node is 0x555555559ae0
Value is 2. Address of the next node is 0x555555559ac0
Value is 1. Address of the next node is 0x555555559b40
The length of the list is 5
[1] + Done                       "/usr/bin/gdb" --interpreter=mi --tty=${DbgTerm}
```

8.Length of the Circular Linked List

```c
void length(){
    struct node *temp=head;
    int counter=0;
    do {
        counter++;
        temp = temp->link;
    } while (temp != head);
    printf("The length of the Linked List is %d\n", counter);
}
```

```
Enter the number of nodes: 4
Enter the data of node 1: 1
Enter the data of node 2: 2
Enter the data of node 3: 3
Enter the data of node 4: 4
Circular Linked List OPERATION MENU
1. FIND LENGTH OF LINKED LIST
2. INSERT NODE TO BEGINNING
3. INSERT NODE TO THE END
4. INSERT NODE TO A SPECIFIC POINT
5. REVERSE THE LINKED LIST
6. DELETE NODE FROM THE BEGINNING
7. DELETE NODE AT THE END
8. DELETE FROM SPECIFIED POSITION
1
The length of the Linked List is 4
[1] + Done                       "/usr/bin/gdb" --interpreter
```