# DATA STRUCTURE AND ALGORITMS

Assignment 01

21ECC212T

Krithika. K S RA2311043010082.

## Circular Linked Lists

→ Creation :

```
#include <stdio.h>
#include <std. lib.h>
struct Node {
    int data ;
    struct Node * next ;
};
void insertAtEnd (struct Node *head. , int data ) {
    struct Node * temp = *head ;
    new_node → data = data ;
    new_node → next = *head ;
    if (*head == NULL) {
        *head = new_node ;
        new_node → next = *head ; }

    else {
        while (temp → next != *head ) {
            temp = temp → next ; }
            temp → next = new_node ;
        }
}
void display (struct Node *head ){
    struct Node * temp = head ;
```

```
return newNode;
void displayList(struct Node * last) {
    if (last == NULL) {
        printf("List is empty");
        return;
    struct Node * temp = last -> next;
    do {
        printf("%d", temp -> data);
        temp = temp -> next;
    }
    while (temp != last -> next);
    printf("(Back to start)\n"); }
    int main () {
        struct Node * last = NULL;
        last = insertAtEnd(last, 10);
        printf("After inserting 10:");
        displayList(last);
        last = insertAtEnd(last, 20);
        printf("After inserting 20:, ");
        display(last);
        last = insertAtEnd(last, 30);
        printf("After inserting 30: ");
        displayList(last);
        return 0;
}
```

```
printf ("After inserting 20: ");
displayList (last);
last = insertAtBeginning (last, 30);
printf ("After inserting 30: ");
displayList (last);
return 0;
}
```

(iii) At the End

```
#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 = newNode;
    return newNode;
}
struct Node* insertAtEnd (struct Node * last, int data) {
    struct Node * newNode = createNode (data);
    if (last == NULL) {
        return newNode;
    }
newNode -> next = last -> next;
last -> next = newNode
```

```
struct Node * newNode insertAtBeginning (struct Node * last,
                                         int data) {
    struct Node * newNode = createNode(data);
    if (last == NULL) {
        return newNode;
    }
    newNode -> next = last -> next;
    last -> next = newNode;
    return last;
}
void displayList (struct Node * last) {
    if (last == NULL) {
        printf("List is empty \n");
        return 0;
    }
    struct Node * temp = last -> next;
    do {
        printf("%d" -> ", temp -> data);
        temp = temp -> next;
    }
    while (temp != last -> next);
    printf("(back to start)\n");
}
int main() {
    struct Node * last = NULL;
    last = insertAtBeginning (last, 10);
    printf("After inserting 10 : ");
    displayList (last);
    last = insertAtBeginning (last, 20);
```

```
    if (head != NULL) {
        do {
            printf("%d", temp -> data);
            temp = temp -> next;
        }
        while (temp != head);
    }
void main() {
    struct Node *head = NULL;
    insertAtEnd(&head, 10);
    insertAtEnd(&head, 20);
    insertAtEnd(&head, 30);
    display(head);
    return 0;
}
```

→ Insertion :

) At the Beginning :

```
#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 = newNode;
    return newNode;
}
```

```
    int data;
    struct Node * next; }
struct Node * deleteBegin (struct Node * last) {
    if (!last) return NULL;
    if (last -> next == last)
    free (last);
     return NULL;
    struct Node * temp = last -> next;
    last -> next = temp -> next;
    free (temp);
    return last; }
    int main () {
        struct Node *last = malloc (size of (struct Node));
        last -> data = 10;
        last -> next = last;
        struct Node * temp = malloc (size of (struct Node));
        temp -> data = 20;
        temp -> next = last -> next;
        last -> next = temp;
        printf ("Before %d -> %d \n"), last -> next -> data,
                last -> data );
        last = deleteBegin (last);
        printf ("After %d \n", last -> data );
        return 0;
```

(ii) At End :

```
#include <stdio.h>
#include <stdlib.h>
struct Node {
    int data;
```

```
struct Node* temp = last -> next;
do {
    printf("%d -> ", temp -> data);
    temp = temp -> next; }
while (temp != last -> next);
printf("(back to start) \n"); }
int main() {
    struct Node * last = NULL;
    last = insertAtPosition (last, 10);
    last = insertAtPosition (last, 20, 2);
    last = insertAtPosition (last, 30, 3);
    printf("Initial list: ");
    displayList(last);
    last = insertAtPosition (last, 15, 2);
    printf("After inserting 15 at position 2: ");
    displayList (last);
    last = insertAtPosition (last, 25, 4);
    printf("After inserting 25 at position 4: ");
    displayList (last);
    last = insertAtPosition (last, 25, 4);
    printf("After inserting 35 at position 6: ");
    displayList (last);
    last = insertAtPosition (last, 40, 8);
    return 0;
```

## Deletion

(i) At the beginning :

```
#include <stdio.h>
#include <stdlib.h>
struct Node {
```

```
if (position == 1)
  {
    newNode -> next = last -> next;
    last -> next = newNode;
    return last;
  }
struct Node * current = last -> next;
last -> next = newNode;
return last; }
struct Node * current = last -> next;
int count = 1;
while (count < position - 1 && current != last) {
    current = current -> next;
    count++; }
if(count < position - 1) {
    printf("position exceeds lists length \n");
    free(Node);
    return last; }
newNode -> next = current -> next;
current -> next = newNode;
if(current = last) {{
    return newNode; }

return last; }
void displayList (struct Node * last) {
  if (last == NULL) { {
      printf ("List is empty \n");
  return 0;
}
```

(iii) At Any Position

```
#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 = newNode;
    return newNode;
}
struct Node *insertAtPosition(struct Node *last, int
                              data, int position){
    if(position < 1) {
        printf("Invalid position \n");
        return last;
    }
struct Node *newNode = creationNode(data);
if (last == NULL) {
    if (position == 1) {
        return newNode;
    }
    else {
        printf("List is empty. Can only insert at 1");
    free(newNode);
    return last;
    }
}
```

```
    struct Node * next;
};
struct Node * deletePos(struct Node * last, int pos) {
    if (!last || pos < 1) return last;
    if (last -> next == last && pos == 1) {
        free(last);
        return NULL;
    if (pos == 1) {
        struct Node * temp = last -> next;
        last -> next = temp -> next;
        free(temp);
        return last; }
struct Node *      = last -> next;
struct Node * prev = NULL;
int i = 1;
while (i < pos && curr != last) {
        prev = curr;
        curr = curr -> next;
        i++; }
if (curr == last) {
        prev -> next = last -> next;
        free(last);
        return prev; }
    prev -> next = curr -> next;
    free(curr);
    return last; }
int main () {
    struct Node * last = malloc(sizeof(struct Node));
    last -> data = 10;
```

```
    struct Node * next; };
struct Node * deleteEnd (struct Node * last) {
    if (!last) return NULL;
    if (last -> next = LAST)  free(last);
    return NULL;
struct Node *temp = last -> next;
while (temp -> next != last) temp = temp -> next;
temp -> next = last -> next;
free (last);
return temp;
int main () {
    struct Node* last = malloc(sizeof(struct Node));
    last -> data = 10;
    last -> next = last -> next;
    last -> next = temp;
    last = temp;
    printf ("Before %d + %d \n");
    last = deleteEnd (last);
    last -> data;
    printf ("After %d \n");
    last -> data;
    return 0;
}
```

(iii) At Specific position :

```
#include <stdio.h>
#include <stdlib.h>
struct Node {
    int data;
```

```
    display (last);
    return 0;
```

Insertion.

(i) At Beginning :

```
#include <stdio.h>
#include <stdio.h>
struct Node {
        int data ;
        struct Node * next *prev;
    };
struct Node * insertBegin (struct Node * last, int data)
{
  struct Node *newNode = malloc (size of (struct Node));
  newNode ⟶ data = data ;
  if (!last) {
  newNode ⟶ next = newNode ;
  newNode ⟶ prev = newNode;
  return newNode;
}
  newNode ⟶ next = last ⟶ next;
  newNode ⟶ prev = last ;
  last ⟶ next ⟶ prev = newNode;
  return last ;
} int main () {
    struct Node * last = NULL;
    last = insertBegin (NULL, 30);
    last = insertBegin (last, 20);
    last = insertBegin (last, 10);
```

```
    newNode -> data = data;
    newNode -> next = newNode;
    newNode -> prev = newNode;
    return newNode; }
    struct Node * insertEnd(struct Node * last, int
                                        data) {
        struct Node * newNode = createNode(data);
    if(!last) return newNode;
    newNode -> next = last -> next;
    newNode -> prev = last;
    last -> next -> prev = newNode;
    last -> next = newNode;
    return newNode;
}
void display (struct Node * last) {
    if (!last) return;
    struct Node * temp = last -> next;
  do {{
      printf ("%d ", temp -> data);
      temp = temp -> next;
  while (temp != last -> next);
      printf ("\n"); }

int main () {
    struct Node * last = NULL;
    last = insertEnd (last, 10);
    last = insertEnd (last, 20);
    last = insertEnd (last, 30);
    printf ("List : ");
```

```
last -> next = last;
struct Node * last = malloc(size of (struct Node));
last -> data = 20;
last -> next = last;
struct Node * last = malloc(size of (struct Node));
temp -> data = 30;
temp -> next = last -> next;
last -> next = temp;
last = temp;
printf("Before : %d -> %d -> %d \n");
last -> next -> data;
last -> next -> next -> data, last -> data;
last -> deletepos(last, 2);
printf("After : %d -> %d \n", last -> next ->
        data, last -> data);

return 0;
```

## Double Circular List

Creation :

```
# include <stdio.h>
# include <stdlib.h>
struct Node {{
        int data;
        struct Node *next;
        struct Node * prev; };
struct Node * create Node (int data) {
    struct Node * new Node = malloc(size of
                                (struct Node));
```

```
        newNode -> next = last -> next;
        newNode -> prev = last;
        last -> next -> prev = newNode;
        return last; }
    free(newNode);
    return NULL;
    if (pos == 1)
        {
            newNode -> next = last -> next;
            newNode -> prev = last;
            last -> next -> prev = newNode;
            return last;
    struct Node * curr = last -> next;
    int i = 1
    while (i < pos - 1 && curr != last) {
        curr = curr -> next;
        i++;
    newNode -> next = curr -> next;
    newNode -> prev = curr;
    curr -> next -> prev = newNode;
    return (curr == last) ? newNode : last;
}
int main () {
    struct Node * last = NULL;
    last = insertpos(last, 10, 1);
    last = insertpos(last, 30, 2);
    last = insertpos(last, 20, 3);
    struct Node * temp = last -> next;
```

```
int main() {
    struct Node *last = NULL;
    last = insertEnd(last, 10);
    last = insertEnd(last, 20);
    last = insertEnd(last, 30);
    struct Node * temp = last->next;
    do {
        printf("%d -> ", temp->data);
        temp = temp->next;
    } while (temp != last->next);
    printf("\n");
    return 0;
```

(iii) At any position :

```
#include <stdio.h>
#include <stdlib.h>
struct Node {
    int data;
    struct Node * next, *prev;
};
struct Node * insertAtPos(struct Node *, last,
        int data, int pos) {
    if (pos<1) return last;
    struct Node *newNode = malloc(sizeof(struct
                                    Node));
    newNode->data = data;
    if (!last) {
        if (pos == 1) {
```

```
struct Node *temp = last -> next;
do {
    printf("%d -> ", temp -> data);
    temp = temp -> next;
} while (temp != last -> next);
    printf("\n");
    return 0;
```

(ii) At End :

```
#include <stdio.h>
#include <stdlib.h>
struct node {
    int data;
    struct Node *next, *prev;
};
struct Node * insertEnd (struct Node *last, int data)
{
    struct Node *newNode = malloc (sizeof (struct Node));
    newNode -> data = data;
    if (! last) {
        newNode -> next = newNode;
        newNode -> prev = newNode;
        return newNode;
    }
    newNode -> next = last -> next;
    newNode -> prev = last;
    last -> next -> prev = newNode;
    last -> next = newNode;
    return newNode;
```