**Assignment – 4**

1. Write a Menu driven C program to accomplish the following functionalities in doubly linked list.

a) Create a doubly linked list.

b) Display the elements of a doubly linked list.

c) Insert a node at the beginning of a doubly linked list.

d) Insert a node at the end of a doubly linked list.

e) Insert a node before a given node of a doubly linked list.

f) Insert a node after a given node of a doubly linked list.

g) Delete a node from the beginning of a doubly linked list.

h) Delete a node from the end of a doubly linked list.

i) Delete a node after a given node of a doubly linked list.

j) Delete the entire doubly linked list.

#include <stdio.h>

#include <math.h>

#include <string.h>

#include <stdlib.h>

#define PI 3.14159

struct node

{

int data;

struct node \*prev;

struct node \*next;

};

struct node \*insertBeginning(struct node \*head, int var)

{

struct node \*temp = (struct node \*)malloc(sizeof(struct node));

temp->data = var;

temp->prev = NULL;

temp->next = NULL;

if (head == NULL)

{

head = temp;

}

else

{

temp->next = head;

head->prev = temp;

head = temp;

}

return head;

}

struct node \*insertEnd(struct node \*head, int var)

{

struct node \*temp = (struct node \*)malloc(sizeof(struct node));

temp->data = var;

temp->next = NULL;

temp->prev = NULL;

if (head == NULL)

{

head = temp;

}

else

{

struct node \*loop = head;

while (loop->next != NULL)

{

loop = loop->next;

}

loop->next = temp;

temp->prev = loop;

}

return head;

}

struct node \*insertBeforeNode(struct node \*head, int var, int addbefore)

{

struct node \*temp = (struct node \*)malloc(sizeof(struct node));

temp->data = var;

temp->next = NULL;

temp->prev = NULL;

if (head == NULL)

{

head = temp;

}

else if (head->data == addbefore)

{

head = insertBeginning(head, var);

}

else

{

struct node \*loop = head;

struct node \*loop1 = head;

while (loop->next->data != addbefore)

{

loop = loop->next;

loop1 = loop1->next;

}

loop1 = loop1->next;

loop->next = temp;

temp->prev = loop;

temp->next = loop1;

loop1->prev = temp;

}

return head;

}

struct node \*insertAfterNode(struct node \*head, int var, int addafter)

{

struct node \*temp = (struct node \*)malloc(sizeof(struct node));

temp->data = var;

temp->next = NULL;

temp->prev = NULL;

if (head == NULL)

{

head = temp;

}

else

{

struct node \*loop = head;

struct node \*loop1 = head;

while (loop->data != addafter)

{

loop = loop->next;

loop1 = loop1->next;

}

loop1 = loop1->next;

if (loop->next == NULL)

{

loop->next = temp;

temp->prev = loop;

return head;

}

loop->next = temp;

temp->prev = loop;

temp->next = loop1;

loop1->prev = temp;

}

return head;

}

struct node \*deleteBeginning(struct node \*head)

{

if (head == NULL)

{

printf("Linked List is Already Empty\n");

}

else if (head->next == NULL)

{

head = NULL;

}

else

{

head = head->next;

head->prev = NULL;

}

return head;

}

struct node \*deleteEnd(struct node \*head)

{

if (head == NULL)

{

printf("Linked List is Already Empty\n");

}

else if (head->next == NULL)

{

head = NULL;

}

else

{

struct node \*loop = head;

while (loop->next->next != NULL)

{

loop = loop->next;

}

loop->next = NULL;

}

return head;

}

struct node \*deleteAfterNode(struct node \*head, int deleteafter)

{

if (head == NULL)

{

printf("Linked List is Already Empty");

}

else

{

struct node \*loop = head;

while (loop->data != deleteafter)

{

loop = loop->next;

}

if (loop->next == NULL)

{

return head;

}

else if (loop->next->next != NULL)

{

struct node \*temp = loop;

temp = temp->next->next;

loop->next = temp;

temp->prev = loop;

}

else

{

loop->next = NULL;

}

}

return head;

}

void display(struct node \*head)

{

printf("Your Linked List:\n");

struct node \*disp = head;

while (disp != NULL)

{

printf("%d ", disp->data);

disp = disp->next;

}

}

int main()

{

struct node \*head = NULL;

while (1)

{

int ch, num, n;

printf("1. Insert at Beginning\n2. Insert at End\n");

printf("3. Insert Before a Node\n4. Insert After a Node\n");

printf("5. Delete from Beginning\n6. Delete from End\n");

printf("7. Delete After a Node\n8. Delete All:\n");

scanf("%d", &ch);

switch (ch)

{

case 1:

printf("Enter Data: ");

scanf("%d", &n);

head = insertBeginning(head, n);

break;

case 2:

printf("Enter Data: ");

scanf("%d", &n);

head = insertEnd(head, n);

break;

case 3:

printf("Enter Data: ");

scanf("%d", &n);

printf("Before which Number? ");

scanf("%d", &num);

head = insertBeforeNode(head, n, num);

break;

case 4:

printf("Enter Data: ");

scanf("%d", &n);

printf("After which Number? ");

scanf("%d", &num);

head = insertAfterNode(head, n, num);

break;

case 5:

head = deleteBeginning(head);

break;

case 6:

head = deleteEnd(head);

break;

case 7:

printf("After which Number? ");

scanf("%d", &num);

head = deleteAfterNode(head, num);

break;

case 8:

head = NULL;

}

display(head);

int x;

printf("\nDo you want to insert/delete more values?\n1. YES\n2. NO:\n");

scanf("%d", &x);

if (x == 2)

{

exit(0);

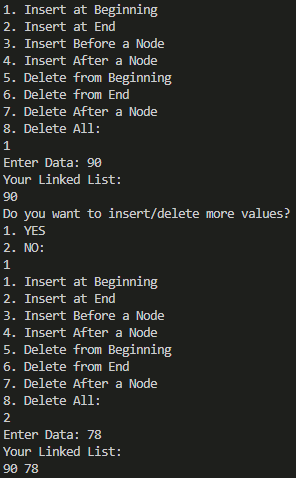
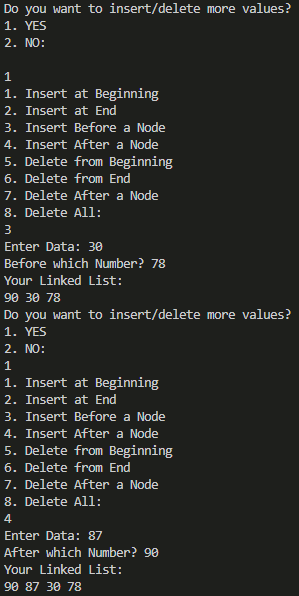
}

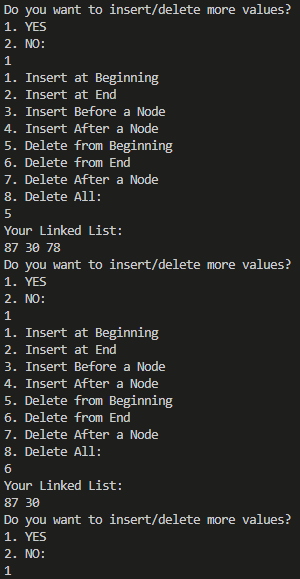
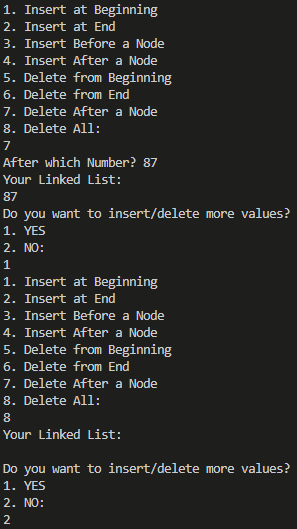
}

return 0;

}

Output:

1. Write a Menu driven C program to accomplish the following functionalities in circular doubly linked list.

a) Create a circular doubly linked list.

b) Display the elements of a circular doubly linked list.

c) Insert a node at the beginning of a circular doubly linked list.

d) Insert a node at the end of a circular doubly linked list.

e) Delete a node from the beginning of a circular doubly linked list.

f) Delete a node from the end of a circular doubly linked list.

g) Delete a node after a given node of a circular doubly linked list.

h) Delete the entire circular doubly linked list.

#include <stdio.h>

#include <math.h>

#include <string.h>

#include <stdlib.h>

#define PI 3.14159

struct node

{

int data;

struct node \*next;

struct node \*prev;

};

struct node \*insertBeginning(struct node \*head, int var)

{

struct node \*temp = (struct node \*)malloc(sizeof(struct node));

temp->data = var;

temp->next = temp;

temp->prev = temp;

if (head == NULL)

{

head = temp;

}

else

{

struct node \*t = head;

t = t->prev;

temp->next = head;

temp->prev = t;

head->prev = temp;

t->next = temp;

head = temp;

}

return head;

}

struct node \*insertEnd(struct node \*head, int var)

{

struct node \*temp = (struct node \*)malloc(sizeof(struct node));

temp->data = var;

temp->next = temp;

temp->prev = temp;

if (head == NULL)

{

head = temp;

}

else

{

struct node \*t = head;

t = t->prev;

t->next = temp;

temp->prev = t;

temp->next = head;

head->prev = temp;

}

return head;

}

struct node \*deleteBeginning(struct node \*head)

{

if (head == NULL)

{

printf("Linked List is Already Empty\n");

}

else if (head->next == head)

{

head = NULL;

}

else

{

struct node \*temp = head;

temp = temp->prev;

head = head->next;

temp->next = head;

head->prev = temp;

}

return head;

}

struct node \*deleteEnd(struct node \*head)

{

if (head == NULL)

{

printf("Linked List is Already Empty\n");

}

else if (head->next == head)

{

head = NULL;

}

else

{

struct node \*temp = head;

temp = temp->prev->prev;

temp->next = head;

head->prev = temp;

}

return head;

}

struct node \*deleteAfterNode(struct node \*head, int deleteafter)

{

if (head == NULL)

{

printf("Linked List is Already Empty\n");

}

else

{

struct node \*loop = head;

struct node \*flag = head;

while (loop->data != deleteafter)

{

if (loop->next == flag)

{

printf("Invalid Number");

return head;

}

loop = loop->next;

}

if (loop->next == head)

{

head = deleteBeginning(head);

}

else

{

struct node \*temp = loop;

temp = temp->next->next;

loop->next = temp;

temp->prev = loop;

}

}

return head;

}

void display(struct node \*head)

{

printf("Your Linked List:\n");

if (head != NULL)

{

struct node \*loop = head;

do

{

printf("%d ", loop->data);

loop = loop->next;

} while (loop != head);

}

}

int main()

{

struct node \*head = NULL;

while (1)

{

int ch, num, n;

printf("1. Insert at Beginning\n2. Insert at End\n");

printf("3. Delete from Beginning\n4. Delete from End\n");

printf("5. Delete After a Node\n6. Delete All:\n");

scanf("%d", &ch);

switch (ch)

{

case 1:

printf("Enter Data: ");

scanf("%d", &n);

head = insertBeginning(head, n);

break;

case 2:

printf("Enter Data: ");

scanf("%d", &n);

head = insertEnd(head, n);

break;

case 3:

head = deleteBeginning(head);

break;

case 4:

head = deleteEnd(head);

break;

case 5:

printf("After which Number? ");

scanf("%d", &num);

head = deleteAfterNode(head, num);

break;

case 6:

head = NULL;

}

display(head);

int x;

printf("\nDo you want to insert/delete more values?\n1. YES\n2. NO:\n");

scanf("%d", &x);

if (x == 2)

{

exit(0);

}

}

return 0;

}

Output:

