Page No.

Date __/__/____

1BM19CS052

LAB PROGRAM -5

```
#include <stdio.h>
#include <stdlib.h>
struct node {
int info;
struct node * link;
};
typedef struct node *NODE;
NODE getnode() {
 NODE x;
x = (NODE) malloc (sizeof (struct node));
if (x == NULL) {
printf ("Memory full\n");
exit (0);
}
return x;
}
NODE insert_front (NODE first, int item) {
 NODE temp;
temp = getnode ();
temp -> info = item;
temp -> link = NULL;
if (first == NULL)
return temp;
temp -> link = first;
first = temp;
return first;
}
NODE delete_front (NODE first) {
 NODE temp;
```

```
if (first == NULL) {
printf ("List is empty cannot delete \n");
return first;
}
temp := first;
temp = temp -> link;
printf ("Item deleted at front end is %d\n",
        first free (first);
return temp;
}.
NODE insert_rear(NODE first, int item) {.
NODE temp, cur;
temp = getnode ();
temp -> info = item;
temp -> link = NULL;
if (first == NULL)
return temp;
cur = first;
While (cur -> link != NULL)
cur = cur -> link;
cur -> link = temp;
return first;
}
NODE delete_rear(NODE first) {.
NODE cur, prev;
if (first == NULL) {.
printf ("List is empty cannot delete \n");
return first;
}
if (first -> link == NULL) {.
printf ("Item deleted is %d\n", first -> info);
```

```
free (first);
return NULL;
}
prev = NULL;
cur = first;
while (cur -> link != NULL) {
prev = cur;
cur = cur -> link;
}
printf ("Item deleted at rear end is %d", cur->
        info);
free (cur);
prev = cur;
cur = cur -> link;
}
printf ("Item deleted at rear end is %d",
        cur -> info);
free (cur);
prev -> link = NULL;
return first;
}
NODE insert_pos (int item, int pos, NODE
                 first) {
NODE temp, cur, prev;
int count;
temp = getnode();
temp -> info = item;
temp -> link = NULL;
if (first == NULL && pos == 1) {
return temp;
}
```

```
if (first == NULL) {
printf ("Invalid position \n");
 return first;
}
if ( pos == 1) {
temp -> link = first;
first = temp;
return temp;
}
count=1;
prev= NULL;
 cur = first;
while ( cur != NULL && count != pos ) {
 prev = cur;
 cur = cur -> link;
 count ++;
}
if ( count == pos ) {
prev -> link = temp;
temp -> link = cur;
return first;
}
 printf ("Invalid position \n");
 return - first;
 }
 NODE delete_pos ( int pos, NODE first ) {
 NODE cur;
 NODE prev;
int count, flag = 0;
if ( first = NULL || pos < 0) {
printf ("Invalid position \n");
return NULL;
```

```
if (pos == 1) {
cur = first;
first = first -> link;
freenode (cur);
return first;
}
prev = NULL;
cur = first;
count = 1;
while (cur != NULL) {
if (count == pos) {
flag = 1;
break;
}
count++)
prev = cur;
cur = cur -> link;
}
if (flag == 0) {
printf ("Invalid position \n");
return first;
}
printf ("Item deleted at given position
        %d \n", cur -> info);
}
}
void main()
{
int item, choice, key, pos;
int count = 0;
NODE first = NULL;
```

```
for(;;){
printf("\n 1: Insert rear \n2: Delete rear
        \n3: Insert front \n4: Delete front
        \n5: Insert info position \n6:
        Delete info position \n7: Display
        list \n8: Exit \n");

printf("Enter the choice: ");
scanf("%d", &choice);
switch(choice){
 case 1: printf("Enter the item at rear
                end\n");
 Scanf("%d", &item);
 first = insert_rear(first, item);
 break;
case 2: first = delete_rear(first);
 break;
case 3: printf("Enter the item at front
                end\n");
 Scanf("%d", &item)
 first = insert_front(first, item);
 break;
case 4: first = delete_front(first);
 break;
case 5: printf("Enter the item to be
                inserted at given position");
 scanf("%d", &item);
 printf("Enter the position\n");
 scanf("%d", &pos);
 first = insert_pos(item, pos, first);
  break;
```

```
    newnode -> next = NULL;
    head = newnode;
    printf("Node created\n");
}
else {
    temp = head;
    while (temp -> next != NULL) {
        temp = temp -> next;
    }
    temp -> next = newnode;
    newnode -> next = NULL;
    printf("Node created at the end.\n");
}
}
void insert_at_first() {
    if (head == NULL) {
        insert_at_end();
        return;
    }
    struct node *newnode;
    int at_end();
    reboom;
}
struct node *newnode;
int ele;
printf("Enter the element to be inserted
        at first position: ");
scanf("%d", &ele);
newnode = (struct node *)malloc(sizeof
        (struct node));
Scanf("%d", &ele);
```

```
newnode = (struct node *) malloc (sizeof
    (struct node));

newnode -> data = ele;
newnode -> next = head;
head = newnode;
printf ("Element inserted at the first
            position of the list.\n");
}

void insert_at_any_pos (int pos) {
if (head == NULL) {
insert_at_first ();
return;
}
if (pos > length()) {
    insert_at_end();
    return;
}
struct node *newnode, *temp;
temp = head;
int ele;
printf ("Enter the element to be
                inserted : ");
scanf ("%d", &ele);
int jump = 1;
while (jump < pos-1) {
temp = temp -> next;
jump++;
}
```

```
newnode = (struct node *) malloc (sizeof
                (struct node));
newnode -> data = ele;
newnode -> next = temp -> next;
temp -> next = newnode;
printf ("Element inserted at
                position %d \n", pos);
}

void display () {
struct node * ptr = NULL;
ptr = head;
if (ptr == NULL)
        printf ("No data to print \n");
else {
        print ("List Contents : \n");
        while (ptr != NULL) {
            printf ("%d", ptr -> data);
            ptr = ptr -> next;
        }
        printf ("\n");
    }
}

int main () {
        int choice;
        int pos;
        printf ("SINGLY LINKED LIST \n");
        printf ("1. Insert at back \t 2. Insert
                at front \t 3. Insert at any
                position \t 4. Display \t 5. Exit");
```

```
printf("Enter your choice : ");
scanf("%d", &choice);

while (choice != 5) {
    switch (choice) {
    case 1: insert_at_end();
            break;
    case 2: insert_at_first();
            break;
    case 3:
            printf("Enter the position you
                    want to insert the
                    new element at : ");
            scanf("%d", &pos);
            if (pos == 1) {
                insert_at_first();
                break;
            }
            insert_at_anypos();
            break;
    case 4: display();
            break;
    }
    printf("1. Insert at back \t 2. Insert
            at front \t 3. Insert at any
            position \t 4. Display \t 5. Exit");

    printf("Enter your choice : ");
    scanf("%d", &choice);
    }
    return 0;
}
```