Program:

```
# include <stdlib.h>
# include <stdio.h>
# include <malloc.h>
# include <string.h>
struct node {
    int id;
    char name[10];
    int sem;
    struct node * next;
};
struct node * start = NULL;
struct node * create (struct node*);
struct node * display (struct node*);
struct node * insert_beg (struct node*);
struct node * insert_end (struct node*);
struct node * insert_bef (struct node*);
struct node * insert_aft (struct node*);
struct node * delete_beg (struct node*);
struct node * delete_end (struct node*);
struct node * delete_node (struct node*);
struct node * delete_aft (struct node*);
struct node * delete_list (struct node*);
int main () {
    int option;
    do {
        printf ("\n1: Create a list ");
        printf ("\n2: Display the list ");
        printf ("\n3: Add a node at the beginning ");
        printf ("\n4: Add a node at end ");
printf ("\n5. Add a node before a given node");
printf ("\n6. Add a node after a given node");
printf (" \n7. Delete a node from beginning");
printf ("\n8. Delete a node from end");
printf (\n9. Delete a ~~node~~ given node);
 printf (\n10. Delete a node after a given node");
printf (\n11. Delete an entire list ");
  printf (\n12. EXIT ");
  printf (\n Enter your choice : ");
  scanf("%d", & option);
  switch (option) {
    case1: start = create (start);
      printf ("\n Linked list Created! ");
      break;
      case2: start = display (start);
      break;
      case 3: start = insert_beg (start);
      ~~case~~ break;
      case 4: start = ~~start~~ insert_end (start);
        break;
      case 5: start = insert_bef (start);
       break;
      case 6: start = insert_aft (start);
       break;
      Case 7: start = delete_beg (start);
        break;
      case 8: start = delete_end (start);
      break;
     case 9: start = delete_node (start);
     break;
     case 10: start = delete_aft (start);
```

```
        break;
      case 11 : start = delete_list ( start );
        printf ( "\n Linked list deleted : " );
        break;
    }
  } while ( option != 12 );
  return 0;
}
struct node * create ( struct node * start ) {
        struct node * ptr, * new_node ;
        int s_id , s_sem;
        char s_name [10];
        printf ( "\n Enter -1 ~~as~~ id to end" );
        printf ( "\n Enter the id: " );
        scanf ( " %d", & s_id );
        printf ( "\nEnter the semester :" );
        scanf ( " %d", & s_sem);
        printf ( " \n Enter the name " );
        scanf ( " %s", s_name: ");
        while ( s_id != -1 ) {
          new_node = ( struct node * ) malloc
                   ( sizeof ( struct node ));
          new node -> id = s_id ;
          new node -> sem = s_sem ;
          strcpy ( new node -> name, s_name );
          if ( start = = NULL ) {
                 new_node -> next = NULL ;
                 start = new_node ;
          }
          else {
                 ptr = start ;
    while ( ptr -> next ! = NULL )
        ptr = ptr -> next ;
        ptr -> next = new_node ;
         new_node -> next = NULL ;
      }
      printf ( " \n Enter the id : " ) ;
      scanf ( " %d " & s_id ) ;
      printf ( " \n Enter the semester : " ) ;
      scanf ( " %d ", & s_sem ) ;
      printf ( " \n Enter the name : " ) ;
      scanf ( " %s ", s_name ) ;
    }
      return start ;
   }
struct node * insert_beg ( struct node * start ) {
      struct node * new_node ;
      int s_id , s_sem ;
      char s_name ;
      printf ( " \n Enter the semester : " ) ;
       scanf ( " %d " s_sem ) ;
       printf ( " \n Enter the name : " ) ;
       scanf ( " %s " , &s_name ) ;
       new_node = ( struct node * ) malloc ( size of ( struct node ) ) ;
       new_node -> id = s_id ;
       new_node -> sem = s_sem ;
        strcpy ( new_node -> name , s_name ) ;
        new_node -> next = start ;
        start = new_node ;
        return start ;
  }
```

```
struct node* insert_end(struct node* start) {
        struct node *ptr, * new_node;
        int s_id, s_sem;
        char s_name;
        printf ("\n Enter the semester :");
        scanf ("%d", & s_sem);
        printf ("\n Enter the name");
        scanf ("%s", s_name);
        new_node = (struct node*) malloc(sizeof (struct node));
        new_node -> id = s_id;
         new_node -> sem = s_sem;
         strcpy (new_node -> name, s_name);
          new node -> next = NULL;
          ptr = start;
         while ( ptr -> next != NULL)
                ptr = ptr -> next;
          ptr -> next = new_node;
          return start;
    }
struct node * insert_bef( struct node* start) {
      struct node * ptr, * preptr, * new_node;
       int s_id, s_sem, pres_id ;
        char s_name;
         printf ("\n Enter the id :");
          scanf(" %d", & s_id);
          printf ("\n Enter the semester :");
          scanf(" %d", & s_sem);
          printf ("\n Enter the name :");
          scanf("%s", s_name);
          printf ("\n Enter the id before which it has
          to be inserted :");
```

```
scanf ("%d", & pres_id );
new_node = (struct node) malloc(size of( struct node));
new_node -> id = s_id ;
new_node -> sem = s_sem ;
strcpy (new_node -> name, s_name );
ptr = start ;
while (ptr -> id != pres_id ) {
        preptr = ptr ;
        ptr = ptr -> next ;
    }
    preptr -> next = new_node ;
    new_node -> next = ptr ;
    return start ;
  }
struct node * insert_aft (struct node * start ) {
    struct node * ptr, * preptr, * new_node ;
    int s_id , s_sem, pres_id ;
    char s_name ;

    printf ("\nEnter an id :");
    scanf( "%d", & s_id );
    printf (" \n Enter the semester.");
    scanf( "%d" & s_sem) ;
    printf(" \n Enter the name :");
    scanf (" %s", s_name );
    printf (" Enter the id after which has to be inster
scanf( "%d" & pres_id );                                  ted ");
new_node =(struct node * ) malloc ( size of ( struct node)),
new_node -> id = s_id ;
new_node -> sem = s_sem ;
strcpy ( new_node -> name , s_name ) ;
ptr = start ;
```

```
preptr = ptr;
while (preptr -> id != pres_id) {
        preptr = ptr;
        ptr = ptr -> next;
    }
    preptr -> next = new_node;
    new_node -> next = ptr;
    return start;
}
struct node * del_end(struct node * start) {
    struct node * ptr, * preptr;
    ptr = start;
    while (ptr -> next != NULL) {
            preptr = ptr;
            ptr = ptr -> next;
        }
        preptr -> next = NULL;
        free (ptr);
        return start;
    }
struct node * delete_~~node~~ (struct node * start) {
    struct node * ptr, * preptr;
    int s_id;
    printf("Enter the id which has to be deleted");
    scanf("%d", & s_id);
    ptr = start;
    if (ptr -> id == s_id) {
        start = delete_beg(start);
        return start;
    } else {
```

```
    while (ptr->id != s_id) {
        preptr = ptr;
        ptr = ptr->next;
    }
    preptr->next = ptr->next;
    free(ptr);
    return start;
   }
  }

struct node * delete_aft(struct node * start) {
     struct node * ptr, * preptr;
     int s_id;
     printf("Enter the id after which the node has to be
     deleted");
     scanf("%d", & s_id);
     ptr = start;
     preptr = ptr;
     while (preptr -> id != s_id) {
         preptr = ptr;
         ptr = ptr -> next;
     }
     preptr -> next = ptr -> next;
     free (ptr);
     return start;
    }
struct node * del_list (struct node * start) {
          struct node * ptr;
          if (start != NULL) {
             ptr = start;
          while (ptr != NULL) {
              start = delete_beg (ptr);
    } }       ptr = start
```