```
#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;
}
void freenode(NODE x) {
   free(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 -> info);
    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->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("Invalic position\n");
      return first;
    }
    printf("Item deleted at given position is %d \n",
                                        cur->info);
    prev->link = cur->link;
    freenode(cur);
    return first;
}
void display (NODE first) {
    NODE temp
    if(first == NULL)
      printf("List empty cannot display items \n");
    for(temp=first; temp != NULL; temp = temp->link){
        printf("%d \t", temp->info);
    }
}
```

```
void main() {
 int item, choice, key, pos;
 int count = 0;
 NODE first = NULL;
 for (;;) {
    printf("\n1. Insert rear\n 2. Delete rear\n
       3. Insert front\n 4. Delete front\n 5. Insert
        into position\n 6. Delete into position\n
        7. Display list\n 8. 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: printf first = delete_rear(first);
             break;
     case 3: printf("Enter 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\n");
             scanf("%d", &item);
             printf("Enter the position:\n");
             scanf("%d", &pos);
             first = insert_pos(item, pos, first);
             break;
```

```
        case 6: printf("Enter the position: \n");
                scanf("%d", &pos);
                first = delete_pos(pos, first);
                break;
        case 7: display(first);
                break;
        default: exit(0)
        break;
    }
  }
}
```