```
#include <stdio.h>
#include <stdlib.h>
#define MAXSIZE 50
int arr[MAXSIZE];
int size = 0;
void display() {
  int i;
  if(size == 0) {
    printf("array is empty\n");
  }else{
    for(i = 0;i < size; i++) {
      printf("%d ", arr[i]);
    }
    printf("\n");
  }
}
void insertAtPosition(int position,int element) {
  int i;
  for(i = size; i > position; i--) {
    arr[i] = arr[i - 1];
  }
  arr[position] = element;
  size++;
}
void deleteAtposition(int position) {
  int i;
  for(i = position; i < size - 1; i++) {
    arr[i] = arr[i + 1];
  }
  size--;
```

```
}
int main() {
  while (1) {
    printf("1. Insert END\n");
    printf("2. Insert Specified Position\n");
    printf("3. Delete Specified position\n");
    printf("4. Display\n");
    printf("5. Exit\n");
    int choice;
    int position;
    int element;
    printf("enter your choice: ");
    scanf("%d", &choice);
    switch(choice) {
      case 1:
      if (size == MAXSIZE) {
        printf("Array is full\n");
        break;
      }
      printf("enter the element to be inserted: ");
      scanf("%d", &element);
      arr[size] = element;
      size++;
      break;
    case 2:
      if (size == MAXSIZE) {
        printf("array is full\n");
        break;
      }
      printf("enter the position");
      scanf("%d", &position);
      if (position < 0) {
        printf("invalid position\n");
        break;
      }
      printf("enter the element to be inserted: ");
      scanf("%d", &element);
      insertAtPosition(position, element);
      break;
      case 3:
        if (size == 0) {
          printf("array is empty\n");
          break;
        }
        printf("enter the position to delete:");
        scanf("%d",&position);
        if (position < 0 || position >= size) {
          printf("invalid position\n");
          break;
        }
        deleteAtposition(position);
        break;
      case 4:
        display();
        break;
      case 5:
        exit(0);
      default:
        printf("invalid choice\n");
    }
```

```
  }
  return 0;
}
```

2.
```
#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 = NULL;
  return newNode;
}
void insertAtEnd(struct Node** head, int data) {
  struct Node* newNode = createNode(data);
  if (*head == NULL) {
    *head = newNode;
    return;
  }
  struct Node* temp = *head;
  while (temp->next != NULL) {
    temp = temp->next;
  }
  temp->next = newNode;
}
void printList(struct Node* head) {
  struct Node* temp = head;
  while (temp !=  NULL) {
    printf("%d -> ", temp->data);
    temp = temp->next;
  }
  printf("NULL\n");
}
int main() {
  struct node* head = NULL;
  insertAtEnd(&head, 10);
  insertAtEnd(&head, 20);
  insertAtEnd(&head, 30);
  insertAtEnd(&head, 40);
  printf("Linked List: ");
  printfList(head);
  return 0;
}
```