CEW LAB # 04 (CODES)

```
//exercise 1
   struct contact* addressbook=(struct contact*) malloc(30 * sizeof(struct
contact) );
   int cont_num=0;
   while (1){
      printf("choose:\n1.to insert a contact\n2.to delete a contact\n3.to
exit.");
      int choice;
      scanf("%d",&choice);
      if (choice == 1){
         (cont_num)++;
         addressbook=realloc(addressbook,100*sizeof(struct contact));
         if (addressbook == NULL){
           printf("memory not allocted");
           break;
         }
         printf("enter name:");
         scanf("%s",addressbook->name);
         printf("enter email:");
         scanf("%s",addressbook->email);
         printf("enter phone number:");
         scanf("%s",addressbook->numbr);
         printf("contact saved successfully");
```

```
    }
    else if (choice == 2) {
      char delnum[50];
      printf("enter phone number to be deleted:");
      scanf("%s",delnum);
      for (int i=0;i<=cont_num;i++){
        if (addressbook->numbr==delnum){
          for (int j=i;j<=cont_num;j++){
            strcpy((addressbook)[j].name, (addressbook)[j + 1].name);
            strcpy((addressbook)[j].email, (addressbook)[j + 1].email);
            strcpy((addressbook)[j].numbr, (addressbook)[j + 1].numbr);


          }
        }


      }


    }
    else if (choice == 3){
        break;
    }
}



free(addressbook);
```

```
//exercise 2
struct Node* list1 = NULL;
struct Node* list2 = NULL;

insertEnd(&list1, 1);
insertEnd(&list1, 3);
insertEnd(&list1, 5);

insertEnd(&list2, 2);
insertEnd(&list2, 4);
insertEnd(&list2, 6);

printf("List 1: ");
printList(list1);

printf("List 2: ");
printList(list2);

struct Node* mergedList = mergeSortedLists(list1, list2);


printf("Merged List: ");
printList(mergedList);


freeList(list1);
```

```
  freeList(list2);
  freeList(mergedList);
  return 0;
}


    //exercise 3
  int count = 1;
  struct linkedlist* head = (struct linkedlist*)malloc(sizeof(struct
linkedlist));
  head->next = NULL;


  while (1) {
      int check = 0;
      printf("Enter 1 to enter data in linked list or 0 to exit: ");
      scanf("%d", &check);


      if (check == 1) {
          struct linkedlist* node = (struct linkedlist*)malloc(sizeof(struct
linkedlist));
          printf("Enter number data for linked list: ");
          scanf("%d", &node->data);


          node->next = head->next;
          head->next = node;
          count++;
      } else {
          break;
    }
  }
  int* array = (int*)malloc(count * sizeof(int));

  struct linkedlist* current = head->next;
  for (int i = 0; i < count; i++) {
      array[i] = current->data;
      current = current->next;
  }

  for (int i = 0; i < count; i++) {
      printf("%d ", array[i]);
  }

  //exercise 4

    struct linkedlist* odd = (struct linkedlist*)malloc(sizeof(struct
linkedlist));
    odd->next = NULL;
    struct linkedlist* curr = odd;

    for (int j = 0; j <= 50; j++) {
        curr->data = j;
        curr->next = (struct linkedlist*)malloc(sizeof(struct linkedlist));
        curr = curr->next;
    }
```

```
curr->next = NULL;


curr = odd;
struct linkedlist* temp;
while (curr->next != NULL && curr->next->next != NULL) {
    temp = curr->next;
    curr->next = curr->next->next;
    free(temp);
    curr = curr->next;
}
curr = odd;
while (curr->next != NULL) {
    printf("%d ", curr->data);
    curr = curr->next;
}
```