Lab report

Palestine Relief Donation Management System

Corresponding Code:

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#define MAX 100

// ---------------------- Array: Donor Registry ----------------------
char donorRegistry[MAX][50];
int donorCount = 0;

void registerDonor(char name[]) {
   if (donorCount < MAX) {
      strcpy(donorRegistry[donorCount++], name);
      printf("Donor '%s' registered successfully.\n", name);
   } else {
      printf("Donor registry is full!\n");
   }
}

void viewRegisteredDonors() {
   printf("\n--- Registered Donors ---\n");
   for (int i = 0; i < donorCount; i++) {
      printf("%d. %s\n", i + 1, donorRegistry[i]);
   }
}

// ---------------------- Queue: Incoming Donations ----------------------
typedef struct {
   char name[50];
   float amount;
} Donation;

Donation donationQueue[MAX];
int front = -1, rear = -1;

void addDonationToQueue(char name[], float amount) {
   if (rear == MAX - 1) {
      printf("Donation queue full!\n");
      return;
   }
   if (front == -1) front = 0;
   rear++;
   strcpy(donationQueue[rear].name, name);
   donationQueue[rear].amount = amount;
   printf("Donation from '%s' of $%.2f added to queue.\n", name,
amount);
}

Donation processDonation() {
   Donation d = {"", 0};
   if (front == -1 || front > rear) {
      printf("No donations to process.\n");
      return d;
   }
   d = donationQueue[front++];
   return d;
}

void viewDonationQueue() {
   if (front == -1 || front > rear) {
      printf("Donation queue is empty.\n");
      return;
   }
   printf("\n--- Pending Donations ---\n");
   for (int i = front; i <= rear; i++) {
      printf("%s: $%.2f\n", donationQueue[i].name,
donationQueue[i].amount);
   }
}

// --------------------- Stack: Processed Donations ---------------------
Donation donationStack[MAX];
int top = -1;

void pushDonation(Donation d) {
   if (top < MAX - 1) {
      donationStack[++top] = d;
   } else {
      printf("Donation stack is full!\n");
   }
}

void viewRecentDonations() {
   if (top == -1) {
      printf("No recent donations.\n");
      return;
   }
   printf("\n--- Recently Processed Donations ---\n");
   for (int i = top; i >= 0; i--) {
      printf("%s: $%.2f\n", donationStack[i].name,
donationStack[i].amount);
   }
}

// --------------------- Linked List: Dynamic Donor List ---------------------
typedef struct DonorNode {
   char name[50];
   struct DonorNode* next;
} DonorNode;

DonorNode* head = NULL;

void addDonorToLinkedList(char name[]) {
   DonorNode* newNode = (DonorNode*)malloc(sizeof(DonorNode));
   strcpy(newNode->name, name);
   newNode->next = head;
   head = newNode;
   printf("Donor '%s' added to dynamic list.\n", name);
}

void deleteDonor(char name[]) {
   DonorNode *temp = head, *prev = NULL;
   while (temp != NULL && strcmp(temp->name, name) != 0) {
      prev = temp;
      temp = temp->next;
   }
   if (!temp) {
      printf("Donor not found.\n");
      return;
   }
   if (!prev)
      head = temp->next;
   else
      prev->next = temp->next;
   free(temp);
   printf("Donor '%s' deleted from dynamic list.\n", name);
}

void viewLinkedListDonors() {
   DonorNode* temp = head;
   printf("\n--- Dynamic Donor List ---\n");
   while (temp != NULL) {
      printf("%s\n", temp->name);
      temp = temp->next;
   }
}
// --------------------- Main Menu ---------------------
int main() {
   int choice;
   char name[50];
   float amount;

   while (1) {
      printf("\n==== Palestine Relief Donation Management ====\n");
      printf("1. Register Donor\n");
      printf("2. View Registered Donors\n");
      printf("3. Add Donation\n");
      printf("4. Process Donation\n");
      printf("5. View Donation\n");
      printf("6. View Recent Donations\n");
      printf("7. Add Donor to List\n");
      printf("8. Delete Donor from List\n");
      printf("9. View Dynamic Donor List\n");
      printf("0. Exit\n");
      printf("Choose an option: ");
      scanf("%d", &choice);
      getchar();

      switch (choice) {
          case 1:
              printf("Enter donor name: ");
              fgets(name, sizeof(name), stdin);
              name[strcspn(name, "\n")] = 0;
              registerDonor(name);
              break;

          case 2:
              viewRegisteredDonors();
              break;

          case 3:
              printf("Enter donor name: ");
              fgets(name, sizeof(name), stdin);
              name[strcspn(name, "\n")] = 0;
              printf("Enter donation amount: $");
              scanf("%f", &amount);
              getchar();
              addDonationToQueue(name, amount);
              break;

          case 4: {
              Donation d = processDonation();
              if (d.amount > 0) {
                  pushDonation(d);
                  printf("Processed donation of $%.2f from %s.\n", d.amount,
d.name);
              }
              break;
          }

          case 5:
              viewDonationQueue();
              break;

          case 6:
              viewRecentDonations();
            break;

        case 7:
            printf("Enter donor name: ");
            fgets(name, sizeof(name), stdin);
            name[strcspn(name, "\n")] = 0;
            addDonorToLinkedList(name);
            break;

        case 8:
            printf("Enter donor name to delete: ");
            fgets(name, sizeof(name), stdin);
            name[strcspn(name, "\n")] = 0;
            deleteDonor(name);
            break;

        case 9:
            viewLinkedListDonors();
            break;

        case 0:
            printf("Thank you for supporting Palestine. Exiting...\n");
            exit(0);

        default:
            printf("Invalid choice. Try again.\n");
      }
    }

    return 0;
}
```