```
/*
Vedant V Yelsangikar
CS531 HW2
G01379948
*/

#include<stdio.h>
#include<string.h>
#include<stdlib.h>
#define MAX_LINE_LENGTH 1000

struct address_t
{
    int octet[4];
    char alias[11];
    struct address_t *next;
};
struct address_t *head = NULL;              //Global variable

/*
   Method to insert Nodes into the List.
*/
struct address_t* insertNode(int *adrs, char *alias){
   struct address_t *node = (struct address_t *)malloc(sizeof(struct address_t));
   for(int i=0;i<4;i++){
      node->octet[i] = adrs[i];
   }
   strcpy(node->alias, alias);
   node->next = NULL;
   if(head==NULL){
      head=node;
   }else{
      node->next=head;
      head=node;
   }
   return head;
 }

/*
   Method to read from the file.
*/
 void readFile(){
    FILE    *textfile;
    char    ch[100],name[30];
    int i[4]={0};

    textfile = fopen("CS531_Inet.txt", "r");
    if(textfile == NULL){
      printf("File cannot be opened \n");
    }else{
      printf("File can be accessed \n");

    while(fgets(ch, 20000, textfile) != NULL) {
         sscanf(ch, "%d.%d.%d.%d %s", i, i + 1, i + 2, i + 3,name);
         insertNode(i,name);
    }
    fclose(textfile);
    }
 }

/*
   Method to Validity of the user input in the list.
*/
  int checkValidity(int *adrs, char *alias){
   int count=0;
   for (int i = 0; i < 4; i++){
      if(adrs[i]<1 || adrs[i]>255){
         printf("Invalid entry pleaese Re-");
         return 0;
      }
   }
      if(strlen(alias)<1 || strlen(alias)>10){
         printf("Invalid entry pleaese Re-");
         return 0;
      }
   return 1;
 }

/*
   Method to check duplicate Nodes.
*/
 int duplicateValues(int *adrs){
   // duplicate
   struct address_t *temp = head;
   int count=0;
      while (temp->next != NULL) {
```

```
         for(int i=0;i<4;i++){
            if(temp->octet[i]==adrs[i]){
               count +=1;
            }
         }
         if(count==4){
            printf("\nAddress already present please Re-");
            return 0;
         }
         temp=temp->next;
    }
    return 1;
 }

/*
   Method to add address - option 1
*/
 void addAddress(){
   int adrs[4]={0};
   char input[100],alias[100];
   printf("Enter the IP addresss\n");
   scanf("%s",input);
   scanf("%s",alias);
   sscanf(input, "%d.%d.%d.%d", adrs, adrs + 1, adrs + 2, adrs + 3);
   int check = checkValidity(adrs, alias);          //Method call to check validity.
   int checkDuplicateValues = duplicateValues(adrs);    //Method call to  check duplicate values.
   if(check == 1 && checkDuplicateValues == 1){
      insertNode(adrs, alias);
      printf("\nvalues : %d.%d.%d.%d %s\n",adrs[0], adrs[1], adrs[2], adrs[3],alias);
   }else{
      addAddress();
   }
 }

/*
   Method to display the list address - option 5
*/
 void displayList(){
   struct address_t *ptr = head;
    while(ptr != NULL)
    {
      printf("\n%d.%d.%d.%d %s\n",ptr->octet[0],ptr->octet[1],ptr->octet[2],ptr->octet[3],ptr->alias);
      ptr = ptr->next;
    }
    return;
 }

/*
   Method to Display aliases for location - option 6
*/
 void diplayAlias(){
    int a,b;
    printf("\nEnter the locations : ");
    scanf("%d.%d",&a,&b);
    struct address_t *ptr = head;

    if(a>255 || b>255){
        printf("\nInvalid number please Re-");
        diplayAlias();
    }else{
        while(ptr != NULL){
            if(ptr->octet[0]==a && ptr->octet[1]==b){
               printf("\n%s",ptr->alias);
            }
            ptr = ptr->next;
        }
    }
}

/*
   Method to search from the list address with the alias - option 2
*/
void searchAddress(){
   struct address_t *ptr = head;
   char key[30];
   int flag=0;
   printf("Enter the name you want to search : ");
   scanf("%s", key);
   while (ptr != NULL){
      if(strcmp(ptr->alias,key)==0){
         printf("\nAddress for %s is %d.%d.%d.%d\n",ptr->alias,ptr->octet[0],ptr->octet[1],ptr->octet[2],ptr->octet[3]);
         flag = 1;
         break;
      }else{
         ptr = ptr->next;
      }
   }
```

```
   if(flag != 1){
      printf("\nNot found please Re-");
      searchAddress();
   }

}

/*
   Method to save address to a file -option 7
*/
void saveAddressToFile(){
   char filename[100];
    printf("\nEnter the file name : ");
    scanf("%s",filename);
    FILE *testfile;
    struct address_t* node = head;
    testfile = fopen (filename, "w");
    if (testfile == NULL)
    {
        fprintf(stderr, "\nUnable to Open the File'\n");
        exit (1);
    }
    while(node!=NULL)
    {
        fprintf(testfile,"%d.%d.%d.%d %s\n",node->octet[0],node->octet[1],node->octet[2],node->octet[3],node->alias);
        node = node->next;
    }
    printf("\nFile successfully saved to %s\n", filename);
    fclose(testfile);
}

/*
   Method to delete value from the list - option 4
*/
void deleteAddress(){
    char delete[30];
    int choice;
    printf("\nEnter the alias name to delete :");
    scanf("%s",delete);
    struct address_t *ptr = head;
    struct address_t *temp;
    while(ptr != NULL){
        if(strcmp(ptr->alias,delete)==0){
            printf("\nAddress for %s is %d.%d.%d.%d\n",ptr->alias,ptr->octet[0],ptr->octet[1],ptr->octet[2],ptr->octet[3]);
            printf("\nAre you sure you want to delete this address ? (1/0)\n");
            scanf("%d",&choice);
            if(choice){
                if(strcmp(head->alias,delete)==0){
                    temp = head;
                    head = (head)->next;
                    free(temp);
                }else{
                    struct address_t *current  = head;
                    while(current->next != NULL){
                        if(strcmp(current->next->alias,delete)==0){
                          temp = current->next;
                          current->next = current->next->next;
                          free(temp);
                          break;
                        }else
                            current = current->next;
                    }
                }
                printf("Address successfully deleted");
            }else{
                printf("Address is not deleted");
            }
            return;
        }else{
            ptr = ptr->next;
        }
    }
    printf("\nAddress not found\n");
    displayList();
    return;
}

/*
   Method to update address given alias - option 3
*/
void updateAddress(){
   struct address_t *ptr = head;
   char key[30];
   int new[4]={0};
   char input[30];
   int flag=0;
   printf("Enter the name you want to Update : ");
```

```
   scanf("%s", key);
   while (ptr != NULL){
      if(strcmp(ptr->alias,key)==0){
         printf("\nAddress for %s is %d.%d.%d.%d\n",ptr->alias,ptr->octet[0],ptr->octet[1],ptr->octet[2],ptr->octet[3]);
         flag = 1;
         break;
      }else{
         ptr = ptr->next;
      }
   }
   if(flag != 1){
      printf("\nNot found please Re-");
      searchAddress();
   }
   printf("\nEnter the new IP address: ");
   scanf("%s", input);
   sscanf(input,"%d.%d.%d.%d",new, new + 1, new + 2, new + 3);
   int check = checkValidity(new, key);     //Method call to check validity
   if(check==1){
   printf("\nNew address is %d.%d.%d.%d",new[0], new[1], new[2], new[3]);
   for(int i=0;i<4;i++){
      ptr->octet[i]=new[i];
   }
 }else{
   updateAddress();
   }
}

/*
   Method to Quit the program - option 8
*/
int quitProgram(){   //Quit method
   return 0;
}

 int main(){       //Main method
   readFile();
   int choice,r=1;
   while(r){
    printf("\nSelect a number for the options below :\n\t1.Add address\n\t2.Look up address\n\t3.Update address\n\t4.Delete address\n\t5.
Display list\n\t6.Display aliases for location\n\t7.Save to file\n\t8.Quit\nEnter your choice : ");
    scanf("%d",&choice);
   switch (choice)
   {
   case 1:
      printf("Add address choice\n");
      addAddress();  //Method call to add Address
      break;
   case 2:
      printf("Look up address\n");
      searchAddress();  //Method call to search Address
      break;
   case 3:
      printf("Update address\n");
      updateAddress();   //Method call to update Address
      break;
   case 4:
      printf("Delete address\n");
      deleteAddress();  //Method call to delete Address
      break;
   case 5:
      printf("Display list\n");
      displayList();   //Method call to display
      break;
   case 6:
      printf("Display aliases for location\n");
      diplayAlias();
      break;
   case 7:
      printf("Save to file\n");
      saveAddressToFile();  //Method call to save to a file
      break;
   case 8:
      printf("Quit\n");
      r = quitProgram();  //Method call to terminate
      break;

   default:
      main();
      break;
   }
   }
 }
```