```c
//
//  main.c
//  project2
//
//  Created by Haolin Wang on 3/19/23.
//
// ICSI333. System Fundamentals,
// Spring 2023,
// TA: Manasa Nannuru
// Me: Haolin Wang
// NetID: HW479779
//

#include <stdio.h>
#include <stdlib.h>
#include <time.h>
#include "p1_lib.h"



struct node {
    float value;
    IEEEstruct rep;
    struct node *next;
};

void insert_node (struct node **h, struct node **t, float v, IEEEstruct w){
    struct node *temp;
    if ((temp = (struct node *)malloc(sizeof(struct node)))==NULL){
        printf("Node allocation failed.\n");
        exit(1);
    }
    temp -> value = v;
    temp -> rep = w;

    temp -> next = NULL;
    if(*h == NULL){
        *h = *t = temp;
    } else {
        (*t)-> next = temp; *t = (*t)->next;
    }

}

void print_pos_file(struct node *h){
    FILE *file = fopen("/Users/haolinwang/Documents/UAlbany/Spring
     2023/ICSI333/Programming Assignment/Project
     2/p2_turnin1/project2/pos.txt", "w");
    if (file==NULL){
        exit(-1);
    }
    if (h==NULL){
        fprintf(file, "The list is empty.%c\n", ' ');
    }
    else {
        printf("Value in the list are: (printList)\n");
        while (h!=NULL){
            int i =0, i1=0;
            fprintf(file, "%f\n", h->value);
            fprintf(file, "sign: %d     exponent", h->rep.sign);
            while(i1<=7){
                fprintf(file, "%d", h->rep.exponent[i1]); fflush(stdout);
                i1++;
            }
            fprintf(file,"   mantissa: %c", ' ');
            while(i<=23){
                fprintf(file, "%d", h->rep.mantissa[i]); fflush(stdout);
                i++;
            }
            fprintf(file, "%c\n",' ');
            fprintf(file, "%f\n", h->value);
            h = h->next;
        }
    }
    fclose(file);
}

void print_pf_file(struct node *h){
    FILE *file = fopen("/Users/haolinwang/Documents/UAlbany/Spring
     2023/ICSI333/Programming Assignment/Project
     2/p2_turnin1/project2/pos_frac.txt", "w");
    if (file==NULL){
        exit(-1);
    }
    if (h==NULL){
        fprintf(file, "The list is empty.%c\n", ' ');
    }
    else {
        printf("Value in the list are: (printList)\n");
        while (h!=NULL){
            int i =0, i1=0;
            fprintf(file, "%f\n", h->value);
            fprintf(file, "sign: %d     exponent", h->rep.sign);
            while(i1<=7){
                fprintf(file, "%d", h->rep.exponent[i1]); fflush(stdout);
                i1++;
            }
            fprintf(file,"   mantissa: %c", ' ');
            while(i<=23){
                fprintf(file, "%d", h->rep.mantissa[i]); fflush(stdout);
                i++;
            }
            fprintf(file, "%c\n",' ');
            fprintf(file, "%f\n", h->value);
            h = h->next;
        }
    }
    fclose(file);
}

void print_neg_file(struct node *h){
    FILE *file = fopen("/Users/haolinwang/Documents/UAlbany/Spring
     2023/ICSI333/Programming Assignment/Project
     2/p2_turnin1/project2/neg.txt", "w");
    if (file==NULL){
        exit(-1);
    }
    if (h==NULL){
        fprintf(file, "The list is empty.%c\n", ' ');
    }
    else {
        printf("Value in the list are: (printList)\n");
        while (h!=NULL){
            int i =0, i1=0;
            fprintf(file, "%f\n", h->value);
            fprintf(file, "sign: %d     exponent", h->rep.sign);
            while(i1<=7){
                fprintf(file, "%d", h->rep.exponent[i1]); fflush(stdout);
                i1++;
            }
            fprintf(file,"   mantissa: %c", ' ');
            while(i<=23){
                fprintf(file, "%d", h->rep.mantissa[i]); fflush(stdout);
                i++;
            }
            fprintf(file, "%c\n",' ');
            fprintf(file, "%f\n", h->value);
            h = h->next;
        }
    }
    fclose(file);
}

void print_nf_file(struct node *h){
    FILE *file = fopen("/Users/haolinwang/Documents/UAlbany/Spring
     2023/ICSI333/Programming Assignment/Project
     2/p2_turnin1/project2/neg_frac.txt", "w");
    if (file==NULL){
        exit(-1);
    }
    if (h==NULL){
        fprintf(file, "The list is empty.%c\n", ' ');
    }
```

```c
    else {
        printf("Value in the list are: (printList)\n");
        while (h!=NULL){
            int i =0, i1=0;
            fprintf(file, "%f\n", h->value);
            fprintf(file, "sign: %d     exponent", h->rep.sign);
            while(i1<=7){
                fprintf(file, "%d", h->rep.exponent[i1]); fflush(stdout);
                i1++;
            }
            fprintf(file,"   mantissa: %c", ' ');
            while(i<=23){
                fprintf(file, "%d", h->rep.mantissa[i]); fflush(stdout);
                i++;
            }
            fprintf(file, "%c\n",' ');
            fprintf(file, "%f\n", h->value);
            h = h->next;
        }
    }
    fclose(file);
}
void print_list(struct node *h){
    if (h==NULL){
        printf("The list is empty.\n");
    }
    else {
        printf("Value in the list are:\n");
        while (h!=NULL){
            int i =0, i1=0;
            printf("%f\n", h->value);
            printf("sign: %d  ", h->rep.sign);
            printf("exponent: ");
            while(i1<=7){
                printf("%d", h->rep.exponent[i1]); fflush(stdout);
                i1++;
            }
            printf("   mantissa: ");
            while(i<=23){

                printf("%d", h->rep.mantissa[i]); fflush(stdout);
                i++;
            }
            printf("\n");
            printf("%f\n", h->value);
            h = h->next;
        }
    }
}
```

```c
int main(int argc, const char * argv[]) {
    int first_arg = 10;
    printf("You have entered %d arguments:\n", argc);
    printf("The default amount of positive numbers (including fractions) will
     be 10.\n");
    printf("And the default file output format is .txt.\n");
    for (int i=0; i< argc; i++) {
        //printf("%s\n", argv[i]);
    }
    if (argc>1){
        printf("%d\n",atoi(argv[1]));
        first_arg=atoi(argv[1]);
        printf("%d\n", first_arg);
        printf("%s\n", argv[2]);
    }

    struct node *head_pos, *tail_pos;
    struct node *head_pf, *tail_pf;
    struct node *head_neg, *tail_neg;
    struct node *head_nf, *tail_nf;

    head_pos = tail_pos = NULL;
    head_pf = tail_pf = NULL;
    head_neg = tail_neg = NULL;
    head_nf = tail_nf = NULL;

    float face=0;
    int count=0, countPos=0;
    srand(time(NULL));
    IEEEstruct r;

    while (countPos<first_arg){
        face = (float)rand()/(float)RAND_MAX *200-100;
        r = IEEEcal(face);
        if (face>0)
            countPos++;
        if(face>1){
            insert_node(&head_pos, &tail_pos, face, r);
        }
        else if((int)face==0 && face>0){
            insert_node(&head_pf, &tail_pf, face, r);
        }
        else if(face<-1){
            insert_node(&head_neg, &tail_neg, face, r);
        }
        else if((int)face==0 && face<0){
            insert_node(&head_nf, &tail_nf, face, r);
        }

        count++;
    }
    print_pos_file(head_pos);
    print_pf_file(head_pf);
    print_neg_file(head_neg);
    print_nf_file(head_nf);


    return 0;
}
```