```c
/*
programmer: Chau Nguyen
tug37553@gmail.com
*/
#include <stdio.h>
#include <stdlib.h>
#include <float.h>
#include <math.h>
#include <string.h>

#define INIT_SIZE 4
#define EPSILON 0.01

/*
    This lab requires you to write functions that manage
    and process double precision data vectors.  The includes,
    defines, Vector struct, function prototypes, skeleton
    functions and test code in main have been included to
    get you started.

    I suggest writing the functions in order
    of appearance below because some functions, either directly
    or indirectly depend on others.  You have to create a vector
    and insert elements before printing a vector.  The sum function
    can be used in avg, avg can be used in var and var in stdv.  The
    dbl_equals function should be used to compare doubles within a
    threshold to account for arithmetic precision errors on doubles.

    Make sure that you check that file open and memory allocation
    is successful before using the file or memory.  Points will be
    deducted if you code can crash in any way.  So be sure to
    consider and handle all possible threats that may crash your
    application.

    This code will be used in your final project to create static
    and dynamic libraries.

    Good luck!
*/


/*
    Struct for a vector of double.  It contains a pointer
    to a vector of double, an integer for allocated length
    of the vector and and integer for the count of double
    inserted.
*/
typedef struct{
    double *vector;
    int count;
    int length;
}Vector;


// Prototypes
Vector create_vector(int length);
Vector empty_vector();
int insert(Vector *vec, double dbl);
void print_vec(Vector vec);
void delete_vector(Vector *vec);
Vector copy(Vector vec);
Vector copy_range(Vector vec, int from, int to);
void clear_vector(Vector *vec);
void zeros(Vector *vec);
```

```c
void fill(Vector *vec, double dbl);
Vector read_from_file(char *filename);
int write_to_file(Vector vec, char *filename);
Vector get_from_con();
void add_from_con(Vector *vec);
void swap(Vector *vec, int i, int j);
void sort(Vector *vec);
void reverse(Vector *vec);
int dbl_equals(double d1, double d2);
int search(Vector vec, double dbl);
double sum(Vector vec);
double avg(Vector vec);
double var(Vector vec);
double stdv(Vector vec);
Vector add(Vector v1, Vector v2);
Vector sub(Vector v1, Vector v2);
Vector mul(Vector v1, Vector v2);
Vector divv(Vector v1, Vector v2);
double dot(Vector v1, Vector v2);
int equals(Vector v1, Vector v2);


// Main
int main(){

    // Test create, insert and print
    printf("Test create, insert and print\n");
    Vector v1 = create_vector(2);
    insert(&v1, 1.0);
    insert(&v1, 2.0);
    insert(&v1, 3.0);
    insert(&v1, 4.0);
    insert(&v1, 5.0);
    print_vec(v1);
    printf("done\n");

    // Test copy and copy range
    printf("Test copy and copy range\n");
    Vector v2 = copy(v1);
    print_vec(v2);
    Vector v3 = copy_range(v1, 1, 3);
    print_vec(v3);
    printf("\n");

    // Test zeros and fill
    printf("Test zeros and fill\n");
    zeros(&v3);
    print_vec(v3);

    fill(&v2, 1.0);
    print_vec(v2);
    printf("\n");

    // Test read and write
    printf("Test read and write\n");
    Vector v4 = read_from_file("array_in.txt");
    print_vec(v4);
    write_to_file(v1, "array_out1.txt");
    printf("\n");

    // Test get from con and add from con
    Vector v5 = get_from_con();
    print_vec(v5);
    add_from_con(&v5);
```

```
    print_vec(v5);
    printf("\n");

    // Test swap, sort, reverse and search
    printf("Test swap, sort, reverse and search\n");
    swap(&v1, 0, 4);
    swap(&v1, 1, 3);
    print_vec(v1);
    sort(&v1);
    print_vec(v1);
    reverse(&v4);
    print_vec(v4);
    int i = search(v4, 6.0);
    printf("Found at %d\n\n", i);

    // Test sum, avg, var and stdv
    printf("Test sum, avg, var and stdv\n");
    printf("Sum = %f\n", sum(v4));
    printf("Avg = %f\n", avg(v4));
    printf("Var = %f\n", var(v4));
    printf("Stdv = %f\n\n", stdv(v4));

    // Test add, sub, mul, divv, dot and equals
    printf("Test add, sub, mul, divv, dot and equals\n");
    Vector v6 = add(v1, v4);
    print_vec(v6);
    Vector v7 = sub(v6, v4);
    print_vec(v7);
    Vector v8 = mul(v1, v4);
    print_vec(v8);
    Vector v9 = divv(v8, v4);
    print_vec(v9);
    double dbl = dot(v1, v4);
    printf("Dot = %f\n", dbl);
    i = equals(v1, v4);
    printf("Equals = %d\n", i);
    i = equals(v1, v1);
    printf("Equals = %d\n", i);

    // Free memory
    delete_vector(&v1);
    delete_vector(&v2);
    delete_vector(&v3);
    delete_vector(&v4);
    delete_vector(&v5);
    delete_vector(&v6);
    delete_vector(&v7);
    delete_vector(&v8);
    delete_vector(&v9);
    return 0;
}

/*

    Single vector functions

*/

/*
    Function to create a vector of double.  It accepts the
    integer length of the vector to be created and returns
    a Vector struct containing the length of the allocated
    vector, the count (initialized to zero) of doubles
    inserted, and the allocated vector.
```

```
*/
Vector create_vector(int length){
    Vector vec;
    if(length>0){
    vec.vector=(double *)malloc(length*sizeof(double));
        if(vec.vector==NULL)
            printf("Error");
    }else vec.vector=NULL;
    vec.length=length;
    vec.count=0;
    return vec;

}


/*
    Create an empty vector with vector pointer set to NULL,
    length set to zero and count set to zero.  This is used
    when something fails during the creation of a Vector or
    other functions that return a Vector.
*/
Vector empty_vector(){
    Vector v;
    v.vector=NULL;
    v.count=0;
    v.length= 0;
    return v;
}


/*
    Inserts a double into a vector of double.  If the
    vector is full (i.e. count == length), the length of
    the vector should be doubled and the data from the old
    vector should be copied and the old vector freed.  If
    the length is zero, use the defined initial size.
    You can use malloc, which requires you to manually copy
    and free the old vector or try realloc.
*/
int insert(Vector *vec, double dbl){
    if(vec->count==vec->length){//vector is full
        int length = 0;
        int i = 0;
        if(vec->length == 0)
            length = INIT_SIZE;
        else
            length = vec->length*2;
        double *temp = (double*)malloc(length*sizeof(double));
        if(temp == NULL){
            printf("Memory was not reallocated.\n");
            return -1;
        }
        for(i=0; i<vec->length; i++)
            temp[i] = vec->vector[i];
        double *temp2 = vec->vector;
        vec->vector = temp;
        vec->length = length;//double the size
        if(temp2 != NULL)
            free(temp2);
    }
    vec->vector[vec->count++]=dbl;
    return 0;
}
```

```c
/*
    Prints the count, length and elements of a vector to
    screen.
*/
void print_vec(Vector vec){
    printf("Length = %d ",vec.length);
    printf("Count = %d ",vec.count);
    printf("Elements :\n");
    int i=0;
    for(i=0; i<vec.count;i++){
        printf("%.2f ",vec.vector[i]);
    }
    printf("\n");
}


/*
    Frees the memory allocated for the vector, and sets
    the count and length to zero.  Make sure not the free
    an empty (NULL) vector.
*/
void delete_vector(Vector *vec){
    if(vec->vector!=NULL)
        free(vec->vector);
    vec->vector=NULL;
    vec->count=0;
    vec->length=0;
}


/*
    Creates a new vector with equal count, length and
    elements and returns the vector.
*/
Vector copy(Vector vec){
    Vector newVec; //= malloc(sizeof(vec));
    newVec.vector=(double *)malloc(vec.length*sizeof(double));
    newVec.length=vec.length;
    newVec.count=vec.count;
    int i=0;
    for(i=0; i<vec.count;i++)
       newVec.vector[i]=vec.vector[i];
    return newVec;
}


/*
    Copies a range from a vector to a new vector and
    returns the new vector.  The count and length should
    be equal to the number of elements copied.  Remember
    to check for invalid operations:
        to less than from
        to less than zero
        from greater than or equal to count
*/
Vector copy_range(Vector vec, int from, int to){
    Vector newVec; //= malloc(sizeof(vec));
    int numberElement=to-from+1;
    newVec.vector=(double *)malloc(numberElement*sizeof(double));
    newVec.length=numberElement;
    newVec.count=0;
    int i=0;
    for(i=from; i<(to+1);i++)
```

```
       newVec.vector[newVec.count++]=vec.vector[i];
    return newVec;
}


/*
    Writes zeros to the elements of a vector and sets
    the count to zero.
*/
void clear_vector(Vector *vec){
    int i=0;
    for(i=0; i<vec->count;i++)
       vec->vector[i]=0;
    vec->count=0;
}


/*
    Writes zeros to a vector's elements.
*/
void zeros(Vector *vec){
    int i=0;
    for(i=0; i<vec->count;i++)
       vec->vector[i]=0;
}


/*
    Fills a vector's elements with the value in dbl.
*/
void fill(Vector *vec, double dbl){
    int i=0;
    for(i=0; i<vec->count;i++)
       vec->vector[i]=dbl;
}


/*
    Reads a vector from a file with one double on
    each line in the file and returns the vector.
    Remember that scanf returns a -1 after reading
    the last element in a file.
*/
Vector read_from_file(char *filename){
    //create a vector
    Vector vec; //= malloc(sizeof(vec));
    vec.vector=(double *) malloc(INIT_SIZE*sizeof(double));
    vec.count=0;
    //read file
    FILE *fp;
    fp= fopen(filename,"r");
    if(fp==NULL){
        printf("File error\n");
        exit(1);
    }
    if(vec.count == 0){
        while (fscanf(fp, "%lf", &vec.vector[vec.count])!=-1){
            if(vec.count>INIT_SIZE)
                vec.vector=(double*)realloc(vec.vector, vec.count*sizeof(double));
            vec.count++;
        }
        vec.length=vec.count;
    }
    printf("Size %d\n",vec.count);
```

```
return vec;
}


/*
    Writes a vector's elements to a file.
*/
int write_to_file(Vector vec, char *filename){

    int len = vec.count;
    // Open file to write
    FILE *fp;
    fp = fopen(filename, "w");
    // Check that file is open
    if(fp==NULL){
        printf("Error\n");
        exit(2);
    }
    int i=0;
    for(i=0; i<vec.count;i++)
            fprintf(fp, "%lf ", vec.vector[i]);
            //putc(vec.vector[i], fp);
 fclose(fp);
 return vec.count;
}


/*
    Creates a new vector and gets the elements from
    keyboard input.  Accepts and inserts doubles into
    the vector until Enter is pressed (without any other
    chars).  Remember that scanf will not work unless a
    required data type is entered.  You should use gets
    and check for '\0', when Enter only is pressed.
*/
Vector get_from_con(){
    Vector vec= create_vector(2);
    char input[10];
    while(1){
        printf("Enter a double (press enter to stop): ");
        gets(input);
        if ( strcmp(input,"\0") == 0){
            printf("Out\n");
            break;
        }else{
        double dbl=atof(input);
        insert(&vec, dbl);
        }
    }
return vec;
}


/*
    Adds elements to a vector and gets the elements from
    keyboard input.  Accepts and inserts doubles into
    the vector until Enter is pressed (without any other
    chars).  Remember that scanf will not work unless a
    required data type is entered.  You should use gets
    and check for '\0', when Enter only is pressed.
*/
void add_from_con(Vector *vec){
    char input[10];
    printf("Insert\n");
```

```
    while(1){
        printf("Enter a double (press enter to stop): ");
        gets(input);
        if ( strcmp(input,"\0") == 0){
            printf("Out\n");
            break;
        }else{
        //printf("%s\n",input);
        double dbl=atof(input);
        insert(vec, dbl);
        }
    }
}


/*
    Swaps two elements in a vector.
*/
void swap(Vector *vec, int i, int j){
    double temp=vec->vector[i];
    vec->vector[i]=vec->vector[j];
    vec->vector[j]=temp;
}


/*
    Sorts a vector.  Can use selection or bubble
    sort.
*/
void sort(Vector *vec){
    int minIndex=0;
    int j=0;
    for(j=0;j<vec->count-1;j++){
        minIndex=j;
        int i=0;
        for(i=j+1;i<vec->count;i++){
            if(vec->vector[minIndex]>vec->vector[i]){
                minIndex=i;
            }
        }
            swap(vec,j,minIndex);
    }
}


/*
    Reverses the elements of a vector.
*/
void reverse(Vector *vec){
    int i=0;
    for(i=0;i<(vec->count/2);i++){
        swap(vec,i,vec->count-i-1);
    }
}


/*
    Checks if two doubles are equal given the defined
    EPSILON.  Remember that there should be a threshold
    for which two do floating point values are considered
    equal due to computation of arithmetic operations.
*/
int dbl_equals(double d1, double d2){
    if(fabs(d1-d2)<EPSILON){
```

```
        return 1;
    }else return -1;
}
/*
    Perform a binary search on a sorted vector and return
    the index of the element if found and -1 if not found.
*/
int search(Vector vec, double dbl){
int lowerBound=0;
int upperBound=vec.count;
int midPoint=0;
do{
    midPoint=lowerBound+(upperBound-lowerBound)/2;
    if(dbl>vec.vector[midPoint]){
        lowerBound=midPoint+1;
    }else if(dbl<vec.vector[midPoint]){
        upperBound=midPoint-1;
    }
}while((dbl!=vec.vector[midPoint])&&(lowerBound<upperBound));
    if(dbl==vec.vector[midPoint]){
            return midPoint+1;
    }else return -1;


}
/*
    Calculate and return the sum of the elements in
    a vector.
*/
double sum(Vector vec){
    double sum=0;
    int i=0;
    for(i=0; i<vec.count;i++){
        sum+=vec.vector[i];
    }
    return sum;
}


/*
    Calculate and return the average of the elements in
    a vector.
*/
double avg(Vector vec){
    double ave=sum(vec)/vec.count;
    return ave;
}


/*
    Calculate and return the variance of the elements in
    a vector.
*/
double var(Vector vec){
    double mean=avg(vec);
    double sum=0;
    int i=0;
    for(i=0; i<vec.count;i++){
        double diff=(vec.vector[i]-mean);
        double power=pow(diff,2);
        sum+=power;
    }
    return sum/vec.count;
}
```

```c
/*
    Calculate and return the standard deviation of the
    elements in a vector.
*/
double stdv(Vector vec){
    double variance=var(vec);
    double standardDeviation=sqrt(variance);
    return standardDeviation;
}

/*

    Multiple vector functions

*/

/*
    Perform an element by element addition of two vectors,
    where v3[i] = v1[i] + v2[i] and return the resulting
    vector.
*/
Vector add(Vector v1, Vector v2){
    if(v1.count != v2.count){
        printf("Vector counts don't match\n");
        return empty_vector();
    }
    Vector vec= create_vector(2);
    int i=0;
    for(i=0; i<v1.count;i++){
        insert(&vec,v1.vector[i]+v2.vector[i]);
    }
    return vec;
}


/*
    Perform an element by element subtraction of one vector from
    another, where v3[i] = v1[i] - v2[i] and return the resulting
    vector.
*/
Vector sub(Vector v1, Vector v2){
    if(v1.count != v2.count){
        printf("Vector counts don't match\n");
        return empty_vector();
    }
    Vector vec= create_vector(2);
    int i=0;
    for(i=0; i<v1.count; i++){
        insert(&vec,v1.vector[i]-v2.vector[i]);
    }
    return vec;
}


/*
    Perform an element by element multiplication of two vectors,
    where v3[i] = v1[i] * v2[i] and return the resulting
    vector.
*/
Vector mul(Vector v1, Vector v2){
    if(v1.count != v2.count){
        printf("Vector counts don't match\n");
```

```
        return empty_vector();
    }
    Vector vec= create_vector(2);
    int i=0;
    for(i=0; i<v1.count; i++){
        insert(&vec,v1.vector[i]*v2.vector[i]);
    }
    return vec;
}



/*
    Perform an element by element division of two vectors,
    where v3[i] = v1[i] / v2[i] and return the resulting
    vector.
*/
Vector divv(Vector v1, Vector v2){
    if(v1.count != v2.count){
        printf("Vector counts don't match\n");
        return empty_vector();
    }
    Vector vec= create_vector(2);
    int i=0;
    for(i=0; i<v1.count; i++){
        insert(&vec,v1.vector[i]/v2.vector[i]);
    }
    return vec;
}



/*
    Perform the calculation of the dot product of two vectors,
    where dbl += v1[i] * v2[i] and return the resulting
    double.
*/
double dot(Vector v1, Vector v2){
    if(v1.count != v2.count){
        printf("Vector counts don't match\n");
    }
    int i=0;
        double result=0;
        for(i=0; i<v1.count; i++){
            result+=v1.vector[i]*v2.vector[i];
            //insert(&vec,v1.result);
        }
    return result;
}
/*
    Perform an element by element comparison of two vectors.
    If for every i, v1[i] == v2[i], and the count is equal
    return 1, otherwise zero.
*/
int equals(Vector v1, Vector v2){
     if(v1.count != v2.count)
        printf("Vector counts don't match\n");
    int i=0;
    int res;
    for(i=0; i<v1.count; i++){
        if(v1.vector[i]!=v2.vector[i]){
            return 0;
            break;
        }
```

```
    }return 1;
}
```