//Q1

10 19 30 40 50

```
//Q2 a+b
int Get_1D_Array (int arr[], int n)
{
        return arr[n];
}
```

```
//Q3

int main() {
FILE* fp = fopen("myfile.bin" , "ab+");
        int size = 0;
        fread(&size,sizeof(size),1,fp);
        fseek(fp, -sizeof(char), SEEK_END);
        for (int i= 0; i < size; i++) {
                char ch;
                fread(&ch,sizeof(ch),1,fp);
                fwrite(&ch,sizeof(char),1,fp);
                printf("%c",ch);
                fseek(fp, -2*(i+1)*sizeof(char)-1, SEEK_END);
        }
        printf("\n");
        fclose(fp);
}
```

//Q4

4.a all the .o files will be created. the static lib and the static main application. the dinamic lib and dynamic app will not be created.
4.b

```
OBJECTS_LIB = power.o basicMath.o trig.o

trig.o: trig.c maMath.h
        $(CC) $(FLAGS) -c trig.c
```

4.c
CC=gcc -> CC = ArielCC

4.d
mains is bigger because of the static linkadge which taked more space. It will also run faster.

//Q5

```
typedef struct {
        int rows;
        int cols;
        int **mat;
} matrix, *pmat;

pmat init_mat(int rows, int cols) {
    pmat new_mat = (pmat) malloc(sizeof(matrix));
    new_mat->rows = rows;
    new_mat->cols = cols;
    new_mat->mat = (int **) malloc(rows*sizeof(int*));
    for (int i = 0; i < rows; i++)
        new_mat->mat[i] = (int *) calloc(cols,sizeof(int));
    return new_mat;
}

void assign_mat(pmat m, int *old_mat[]) {
    for (int i = 0; i < m->rows; i++)
        for (int j = 0; j < m->cols; j++)
```

```
            m->mat[i][j] = old_mat[i][j];
}

pmat add_mat(pmat a, pmat b) {
    pmat new_mat = init_mat(a->rows, a->cols);
    for (int i = 0; i < a->rows; i++)
        for (int j = 0; j < a->cols; j++)
            new_mat->mat[i][j] = a->mat[i][j]+b->mat[i][j];
    return new_mat;
}

void print_mat(pmat m) {
    for (int i = 0; i < m->rows; i++) {
        for (int j = 0; j < m->cols; j++)
            printf("%d ",m->mat[i][j]);
    printf("\n");
    }
}

void dest_mat(pmat m) {
    for (int i = 0; i < m->rows; i++)
        free(m->mat[i]);
    free(m->mat);
    free(m);
}

int main() {
    int *a[3];                                      // An array of arrays, to fit "assign_mat"'s definition, which expects an int *[]
    int mm[][4] = {{0,1,2,3},{4,5,6,7},{7,8,9,10}};  // regular 2D array
    a[0] = (int *) &mm[0];
    a[1] = (int *) &mm[1];
    a[2] = (int *) &mm[2];                          // now the array a contains same as the 2D array mm, but in the form "assign_mat"
requires
    pmat mat = init_mat(3,4);
    assign_mat(mat,a);
    print_mat(mat);
    pmat new_mat = add_mat(mat,mat);
    print_mat(new_mat);
    dest_mat(mat);
    dest_mat(new_mat);
    return 0;
}
```