```
#include <stdio.h>
#include <stdlib.h>

/*Aluno: Enzo Gurgel Bissoli, login egb2*/
typedef struct{
    float pReal;
    float pImag;

} Complexo;

void empilhe(Complexo **cPilha, Complexo umC, int *tPilha);
Complexo desempilhe(Complexo **cPilha, int *tPilha);
Complexo topo(Complexo *cPilha, int tPilha);
int pilhaVazia(Complexo *cPilha, int tPilha);
void salvePilha(Complexo *cPilha, int tPilha);
Complexo *recuperePilha(int *tPilha);
void teste(void* t_pointer);
void testep(void** t_pointer);
char* leitura(char buffer, int tam, FILE* stdin);
void** realocapp(void** pointerpp, int op);
void* realocap(void* pointerpp, int op);
void liberaRAM(void** pointerpp, int tamp, int flag);
int novoNum();

int main(void) {
    char buffer[40];
    Complexo **cPilha=NULL;
    Complexo umC;
    Complexo tempC1;
    Complexo tempC2;
    Complexo* teste1 = NULL;
    Complexo* teste2 = NULL;
    int* tPilha = NULL;
    int opc, choice;
    teste((void*)teste1);
    teste((void*)teste2);
    testep((void**)cPilha);
    teste((void*)tPilha);
    *tPilha = 0;
    while(1){
        if(!choice){
        printf("Digite as coordenadas real e imaginaria, nessa
ordem.\n");
         fscanf(leitura(buffer,40,stdin),"%f",umC.pReal);
        fscanf(leitura(buffer,40,stdin),"%f",umC.pImag);
        }
         printf("Digite escolha de 1-7\n");
        fscanf(leitura(buffer,1,stdin),"%d",&opc);
        switch (opc)
        {
      case 1:
       empilhe(cPilha, umC, tPilha);
        choice = novoNum;
       break;
        case 2:
        desempilhe(cPilha, tPilha);
```

```
         choice = novoNum;
           break;
        case 3:
         topo(*cPilha, *tPilha);
         choice = novoNum;
             break;
         case 4:
         pilhaVazia(*cPilha, *tPilha);
         choice = novoNum;
             break;
         case 5:
         salvePilha(*cPilha, *tPilha);
         choice = novoNum;
             break;
         case 6:
         recuperePilha(tPilha);
         choice = novoNum;
             break;
         case 7:
         int count = 1;
             do{
               if(pilhaVazia(*cPilha, *tPilha)){
                  printf("Tudo certo com a pilha inicial\n");
                  if(count==1){
                  umC.pImag=2.5;
                 umC.pReal=3.5;
                 empilhe(cPilha, umC, tPilha);
                 umC.pImag=-1.5;
                 umC.pReal=2.0;
                  }
                  else{
                      desempilhe(cPilha, tPilha);
                      desempilhe(cPilha, tPilha);
                      umC.pImag=-1.5;
                      umC.pReal=2.0;
                     empilhe(cPilha, umC, tPilha);
                     umC.pImag = 2.5;
                     umC.pReal = 3.5;
                  }
                 empilhe(cPilha, umC, tPilha);
                     tempC1=topo(*cPilha, *tPilha);
                     if(tempC1.pImag == umC.pImag && tempC1.pReal ==
umC.pReal){
                         if(count==1){
                          salvePilha(*cPilha, *tPilha);
                          desempilhe(cPilha, tPilha);
                          desempilhe(cPilha, tPilha);
                          teste1 =recuperePilha(*tPilha);
                         }
                         else{
                         teste2=recuperePilha(*tPilha);
                         tempC2=topo(*cPilha, *tPilha);
                         printf("%d",(tempC1.pImag==tempC2.pImag &&
tempC1.pReal==tempC2.pReal)?1:0);
                         }
                     }
                }
```

```
            }while(count--);
        choice = novoNum;
            break;
        default:
            printf("encerrado");
        }
    }
    liberaRAM(cPilha, *tPilha, 1);
    free(tPilha);
    free(teste1);
    free(teste2);
    return 1;
}

int novoNum(){
    char buffer[1];
    int escolha;
    printf("Gostaria de adicionar outro complexo? Se sim digite 0\n");
    fscanf(buffer,"%d",&escolha);
    return escolha;
}

void** realocapp(void** pointerpp, int op){
    realloc(pointerpp, sizeof(pointerpp)+op*sizeof(void**));
    teste(pointerpp);
    return pointerpp;
}
void* realocap(void* pointerp, int op){
    realloc(pointerp, sizeof(pointerp)+op*sizeof(void*));
    teste(pointerp);
    return pointerp;
}

char* leitura(char buffer, int tam, FILE* stdin){
    return fgets(buffer, tam, stdin);
}

void liberaRam(void** pointerpp, int tam, int flag){
    int count;
    for(count=0;count<tam;count++)
        free(pointerpp[tam]);
    if(pointerpp != NULL && flag)
        free(pointerpp);
}

void teste(void* t_pointer){
    if(t_pointer == NULL)
        exit(1);
}

void testep(void** t_pointer){
    if(t_pointer == NULL)
        exit(1);
}

void empilhe(Complexo **cPilha, Complexo umC, int *tPilha){
    cPilha = (Complexo**) realocapp((void**) cPilha, 1);
```

```
    cPilha[*tPilha] = &umC;
    *tPilha = *tPilha++;
}

Complexo desempilhe(Complexo **cPilha, int *tPilha){
    Complexo temp;
    temp = *cPilha[*tPilha];
    liberaRAM(cPilha, 1, 0);
    cPilha = (Complexo**) realocapp((void**) cPilha, -1);
    *tPilha = *tPilha--;
    return temp;
}

Complexo topo(Complexo *cPilha, int tPilha){
    return cPilha[tPilha];
}

int pilhaVazia(Complexo *cPilha, int tPilha){
    return ((cPilha == NULL || tPilha == 0)?1:0);
}

void salvePilha(Complexo *cPilha, int tPilha){
    FILE* pfile = fopen("pilha.bin", "wb");
    teste((void*) pfile);
    fwrite(&tPilha, sizeof(int),1,pfile);
    fseek(pfile, sizeof(int), SEEK_CUR);
    while(tPilha){
        fwrite(&cPilha[tPilha],sizeof(Complexo),1,pfile);
        fseek(pfile, sizeof(Complexo),SEEK_CUR);
        tPilha--;
    }
    fclose(pfile);
}

Complexo *recuperePilha(int *tPilha){
    FILE *pfile = fopen("pilha.bin", "rb");
    Complexo* temp = NULL;
    int size = *tPilha;
    teste((void*) temp);
    temp = (Complexo*) realoca(temp, *tPilha);
    teste((void*) pfile);
    while(size){
        fseek(pfile,sizeof(int),SEEK_CUR);
        fread(&temp[size],sizeof(Complexo),1,pfile);
        size--;
    }
    fread(&size,sizeof(int),1,pfile);
    *tPilha = size;
    fclose(pfile);
    return temp;
}
```