Atividade 01-

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

typedef struct{
int topo;
int *dados;
int capacidade;
} estrutura_Pilha;
void criaPilha(estrutura_Pilha *p, int c){
p->capacidade = c;
p->dados = malloc(p->capacidade * sizeof(estrutura_Pilha));
p->topo = 0;
}
void empilha(estrutura_Pilha *p, int A){
int n_elementos = (p->capacidade * sizeof(estrutura_Pilha))/sizeof(int);
if (n_elementos < p->topo){
p->dados[p->topo]=A;
p->topo=p->topo+1;
}else{
printf("Limite da pilha excedido !");
}
}
void empilhaLaco(estrutura_Pilha *p, int A){
int tamanho = sizeof(int);
int n_elementos = (p->capacidade * sizeof(estrutura_Pilha))/tamanho;
int cond = 1;
int cont = tamanho;
for(int i =0; i<n_elementos; i++){
empilha(p, A);
printf("Tamanho: %d\n", p->topo);
}
}
int desempilha (estrutura_Pilha *p){
if(p->topo ==0){
printf("A pilha está vazia !");
return -1;
}else{
p->topo=p->topo-1;
return p->dados[p->topo];
}
}
int imprimeTamanho(estrutura_Pilha *p){
printf("Tamanho: %d\n", p->topo);
}
int imprimeTopo(estrutura_Pilha *p){
printf("Topo: %d\n", p->dados[p->topo - 1]);
}
int main(){
estrutura_Pilha pilha1;
```

```
criaPilha(&pilha1, 17);
empilhaLaco(&pilha1, 80);
imprimeTamanho(&pilha1);
desempilha(&pilha1);
imprimeTopo(&pilha1);

free(pilha1.dados);
}
```

Atividade 02

01- Complexidade O(F(n))

```
void inserePos(estrutura_Lista *l, int p, int v){
l->dados--;
l->ultimo++;
for(int i = 0; i < p; i++){
l->dados[i] = l->dados[i+1];
}
l->dados[p] = v;

}
```

02- Complexidade O(F(1))

```
}
int retornaTamanho(estrutura_Lista *l){
return l->ultimo;
}
```