**Estrutura de Dados – Lista 1**

Nome: Henrique Servidoni de Oliveira

RA: 2760481821017

**1)** exercicio1.c

**2)** O ***\**** no código significa que a variável ***y*** é um ponteiro, ou seja, ela aponta para o endereço de memória que está sendo passado como parâmetro.

Na linha ***\*y=x+x;*** o valor da variável para qual ***y*** aponta recebe o valor de ***x+x***.

Caso não fosse usado, ***y*** receberia o valor de ***x+x***, que seria desalocado no final da função, não alterando o valor da variável na main.

**3)** Listas são um conjunto de structs, usadas para armazenar vários tipos de dados, como por exemplo informações sobre uma única pessoa. Fazendo uma analogia com uma pilha de questionários: cada informação seria uma das várias respostas do questionário, uma folha de questionário inteira seria uma struct considerada um elemento, ou registro, e a pilha com todos os questionários seria a lista propriamente dita. Uma lista pode ser estática ou dinâmica, a diferença está em como ela está armazenada na memória, enquanto uma estática é um vetor de structs, seu uso é limitado caso uma lista tenha que ultrapassar o limite do vetor ou tenha que ser muito grande ou desperdiçando memória caso seja pequena e o limite grande, a dinâmica usa a memória conforme os elementos vão sendo adicionados e desalocando a cada elemento deletado, um elemento aponta para o outro uma vez que é usado ponteiros e nesse caso, 100% da memória necessária é usada.

Pilhas e Filas são apenas maneiras com que listas são organizadas com relação a adição e remoção de novos elementos. Pilhas, assim como em uma pilha de pratos, os elementos vão sendo adicionados em cima (no início) e as de cima (do início) que são removidas. E filas, os elementos são adicionados por último, e o primeiro a ser adicionado que terá prioridade na remoção.

**4)** exercicio4.c

**5)** Busca Binária:

int Busca\_Binaria(LISTA\* l1; TIPOCHAVE ch) {

int inf, sup, meio;

if(l1->ultimo > MAX){

printf(“Erro na busca”);

return -1;

}

else{

inf = l1->primeiro;

sup = L->ultimo-1;

while(inf <= sup){

meio = (inf + sup) / 2;

if (l1->A[meio].chave == ch){

inf = sup +1;

}

else if(l1->A[meio].chave < ch){

inf = meio + 1;

}

else{

sup = meio - 1;

}

}

}

return meio;

}

**6)** exercicio6.c