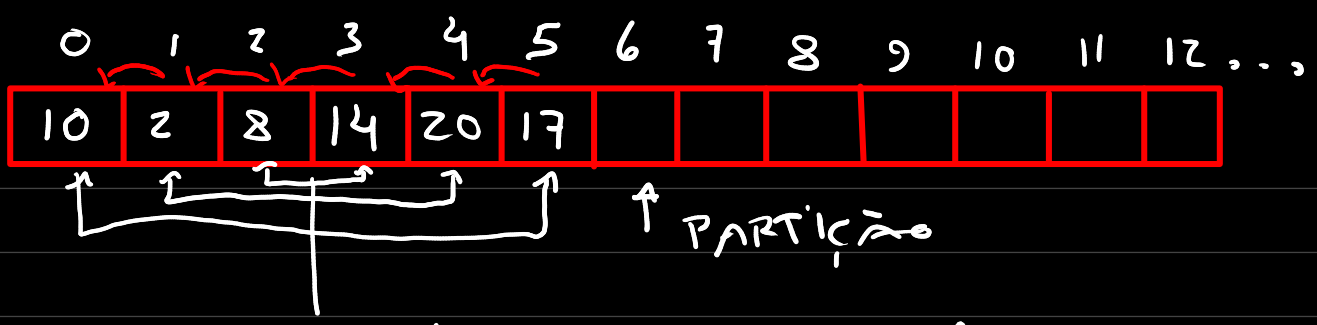
1:




```
i = 0; i < M/2 ; i++
j = M - i - 1;
```

```
A[i] <-> A[j]
T = A[i]
A[i] = A[j]
A[j] = T
```

2:

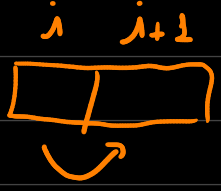
```
lista segunda(lista input) {
        int soma = 0;
        lista output = NULL;
        lista temp = input;
        while(temp != NULL) {
                soma+=temp->valor;
                output = inserir_fim(output, soma);
                temp = temp->prox;
        }
        return output;
}}
```

4:

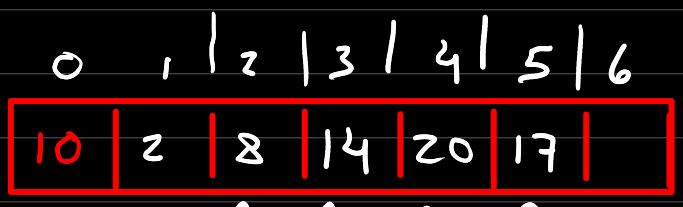
```
void nome_cidades(struct ls_cidade *cidades) {
        int i;
        for(i=0; i<cidades->particao;i++) {
                if( cidades->elementos[i].x > 0 &&
                    cidades->elementos[i].y > 0)
                printf("%s\n", cidades->elementos[i].nome);
        }


}
```

5?

```
void inserir_inicio(struct ls_cidade *cidades,
                                    struct cidade c){

        int i;

        for(i = cidades->particao-1; i > 0; i--) {
          cidades->elementos[i+1] = cidade->elementos[i];
        }

        cidades->elementos[0] = c;
        cidades->particao++;
        }
```

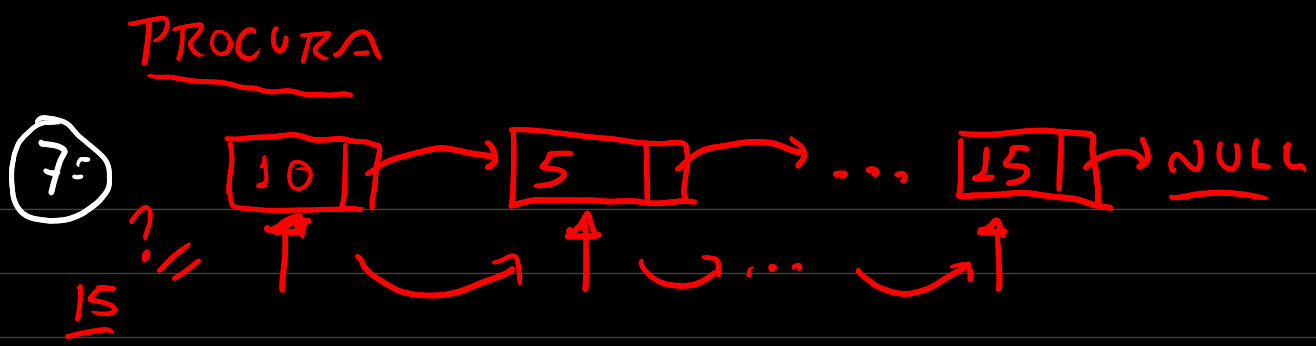



6:




MAIOR = ~~10~~ 14 20

```
void mais_proxima(lista cidades) {
        struct cidade menor;
        lista temp=cidades;
        menor = cidades->valor;

        while(temp != NULL) {
                if(distancia_origem(temp->valor)
                        < distancia_origem(menor)) {
                        menor = temp->valor;
                }

                temp = temp->prox;
        }

        printf("%s", menor.nome);
}

float distancia_origem(struct cidade c) {
        float distancia = sqrt(c.x * c.x + c.y*c.y);
        return distancia;
```




```
float distancia (struct cidade c1, struct cidade c2) {}

struct cidade * procura(char *nome, lista cidades) {
        lista temp;
        temp = cidades;
        while(temp != NULL) {
                if(strcmpi(nome, temp->valor.nome) == 0)
                        return &(temp->valor);
                temp = temp->prox;
        }

        return NULL;
}

float distancia_cidades(char *n1, *n2, lista cidades) {
        struct cidade *c1, *c2;
        c1 = procurar(n1, cidades);
        c2 = procurar(n2, cidades;
        if(c1 != NULL && c2 != NULL)
                return distancia(*c1, *c2);
        return -1;


}
```