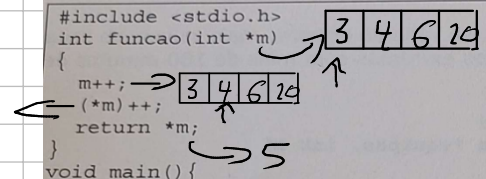
Resolução Exame de AP

1a)

a) Descreva o que faz o programa:

```
#include <stdio.h>
int funcao(int *m)
{
  m++;
  (*m)++;
  return *m;
}
void main(){
  int v[4]={3,4,6,10};
  printf("\n%d",funcao(v));
}
```



(*m)++ é o conteudo de m ++, neste caso 4++=5

Resposta: O programa vai enviar um ponteiro do vetor para a função.
Posteriormente o apontador vai apontar para a 2ª posição do vetor.
Depois o conteudo da 2ª posição do vetor vai incrementar, neste caso o 4++, que resultará no 5 que será o valor apresentado no ecrã

1b)

b) Considere a estrutura:

```
typedef struct ponto{
   int x,y;
}PONTO;
```

e o excerto de código:

```
PONTO *p,ponto={0,0};
p = &ponto;
printf("(%d, %d)\t", (*p).x, p->y);
ponto.x = 10;
p->y = 20;
printf("(%d, %d)\n", p->x, ponto.y);
```

O código não aparenta erros.

Será apresentado (0,0) no 1º printf e (20, 20) no 2º printf.

2.

2. (2.0V) Elabore uma função que, dada uma string, devolve outra, mas sem as vogais.

char * stringSemVogais(char * str)

```
char *stringSemVogais(char *str){
    while(*str != '\0'){
        if(*str == 'a' || *str == 'e' || *str == 'i' || *str == 'o' || *str == 'u'||
         *str == 'A' || *str == 'E' || *str == 'I' || *str == 'O' || *str == 'U')

        {
            *str = ' ';
        }
        str++;
    }
}
```

3.

3. (2.0V) *Elabore uma função que, dado um vetor com N elemento inteiros, calcule e devolva a maior diferença existente entre os seus elementos.*

```
// @param v: Vetor
// @param N: Número de elementos do vetor anterior
                    int maiorDiferenca(int v[],int N)
```

```c
int maiorDiferenca(int v[], int N){
    int i, maior = 0, menor = 0;
    for(i = 0; i < N; i++){ //ciclo para percorrer o vetor
        if(v[i] > maior){   //ver qual o maior valor do vetor
            maior = v[i];
        }
        if(v[i] < menor){ //ver qual o menor valor do vetor
            menor = v[i];
        }
    }
    return maior - menor;  //retorna a diferença entre o maior e o menor valor
}
```

4)

a)

a) (2.5V) Implemente uma função que devolva a valia de uma equipa.

```
// @param equipa: equipa para a qual se pretende determinar a valia
                    float valiaEquipa(Equipa equipa)
```

```c
float valiaEquipa(EQUIPA equipa){
    float valia = 0;
    for(int i = 0; i < MAX; i++){
        if(strcmp(equipa.atletas[i].pos, "Base") == 0){
            valia += 3*equipa.atletas[i].mPontos + 3*equipa.atletas[i].mAssist + 1*equipa.atletas[i].mRessalt - 3*equipa.atletas[i].mPerdas;
        }
        else if(strcmp(equipa.atletas[i].pos, "Extremo") == 0){
            valia += 4*equipa.atletas[i].mPontos + 2*equipa.atletas[i].mAssist + 2*equipa.atletas[i].mRessalt - 3*equipa.atletas[i].mPerdas;
        }
        else if(strcmp(equipa.atletas[i].pos, "Poste") == 0){
            valia += 3*equipa.atletas[i].mPontos + 1*equipa.atletas[i].mAssist + 3*equipa.atletas[i].mRessalt - 3*equipa.atletas[i].mPerdas;
        }
    }
    return valia;
}
```

b)

b) (2.5V) Desenvolva uma função que devolva a valia e o nome da equipa menos valiosa. Deve usar a função anterior para calcular a valia de uma equipa.

```
// @param equipas: Vetor de equipas
// @param n: Número de equipas no vetor anterior
// @param nomeEquipa: Nome da equipa menos valiosa
    float equipaMenosValiosa(Equipa *equipas, int n, char *nomeEquipa)
```

```c
float equipaMenosValiosa(EQUIPA *equipas, int n, char *nomeEquipa){
    float valia = 0;
    float valiaMenor = 999999999999999; //valor muito grande para ter a certeza que a primeira equipa vai passar a ser a equipa menos valiosa na 1 iteração
    int i,j;
    for(i = 0; i < n; i++){
        for(j = 0; j < MAX; j++){
            valia += valiaEquipa(equipas[i]);
        }
        if(valia < valiaMenor){ // se a valia for menor que a valiaMenor a valiaMenor passa a ser a valia
            valiaMenor = valia;
            strcpy(nomeEquipa, equipas[i].nome);
        }
}
}
```

c)

c) (2.5V) Elabore uma função que escreva o nome e a valia do extremo mais valioso de cada equipa. Para esse efeito, só devem ser considerados extremos com mais de 100 minutos de jogo.

```
// @param equipas: Vetor de equipas
// @param n: Número de equipas no vetor anterior
void extremoMaisValioso(Equipa *equipas, int n)
```

```
void extremoMaisValioso(EQUIPA *equipas, int n){
    //extremo mais valioso de cada equipa
    int i, j;
    float valia, maxValia;
    int maxValiaIndex;
    for(i = 0; i < n; i++){
        maxValia = 0;
        maxValiaIndex = 0;
        for(j = 0; j < MAX; j++){
            if(strcmp(equipas[i].atletas[j].pos, "Extremo") == 0){
                valia = 4*equipas[i].atletas[j].mPontos + 2*equipas[i].atletas[j].mAssist + 2*equipas[i].atletas[j].mRessalt - 3*equipas[i].atletas[j].mPerdas;
                if(valia > maxValia){
                    maxValia = valia;
                    maxValiaIndex = j;
                }
            }
        }
        printf("Extremo mais valioso da equipa %s: %s  ", equipas[i].nome, equipas[i].atletas[maxValiaIndex].nome);
}
}
```

d)

d) (2.5V) Desenvolva uma função que que escreva, num ficheiro de texto, os nomes, posições e os tempos de jogo dos atletas de uma determinada equipa.

```
// @param equipas: Vetor de equipas
// @param n: Número de equipas no vetor anterior
// @param nomeEquipa: Nome da equipa
// @param f: ficheiro a gravar
void gravarEquipa(Equipa *equipas, int n, char *nomeEquipa, FILE *f)
```

```
void  gravarEquipa(EQUIPA *equipas, int n, char *nomeEquipa, FILE *f){
    //escrever num ficheiro de texto os nomes, posições e os tempos de jogo dos atletas de uma equipa
    //nomeEquipa é o nome da equipa

    int i;
    for(i=0; i<n; i++){
        if(strcmp(equipas[i].nome, nomeEquipa)==0){
            for(int j=0; j<MAX; j++){
                fprintf(f, "%s %s %d    ", equipas[i].atletas[j].nome, equipas[i].atletas[j].pos, equipas[i].atletas[j].tempo);
            }
        }
    }
}
```

5)

5. (2.0V) Considere a função aux. Explique o funcionamento da função quando se chama aux(3,L,0,5), em que L={-2,-2,1,3,5,6,7,8}.

```
int aux(int x, int L[ ],int inic, int fim){
  int meio = (inic+fim)/2;
  if (inic > fim)
    return 0;
  else {
    if (x == L[meio])
      return 1;
    else{
      if (x > L[meio])
        return aux(x, L, meio+1, fim);
      else
        return aux(x, L, inic, meio-1);
      }
    }
}
```

Este algoritmo é uma implementação recursiva do algoritmo de procura binária. A função leva quatro parâmetros: um inteiro x que representa o elemento que está a ser procurado, uma matriz inteira L que representa o vetor que está a ser pesquisado e os inteiros inic e fim que representam os índices de início e fim do subvetor que está a ser pesquisado.

Quando a função é chamada com os parâmetros aux(3,L,0,5) e L = {-2,-2,1,3,5,6,7,8}, o algoritmo começa a encontrat o meio do subvetor que está a ser pesquisado (meio = (0+5)/2 = 2), ja que é um inteiro. Ele então verifica se o valor neste índice é igual a x (3). Como é, a função retorna 1, indicando que o elemento foi encontrado na lista.

6)

6. (2.0V) Elabore uma função que calcule, de forma eficiente, o valor do somatório $\sum_{i=1}^{N} \frac{i!}{3^{i-1}}$

```c
float somatorio(int N){

    int i, j, fatorial, potencia;
    float soma = 0;
    for(i = 1; i <= N; i++){
        fatorial = 1;
        potencia = 1;
        for(j = 1; j <= i; j++){
            fatorial *= j;
        }
        for(j = 1; j <= i-1; j++){
            potencia *= 3;
        }
        soma += (float)fatorial/potencia;
    }
}
```