Universidade Federal do Maranhão
Departamento de Informática
Ciência da Computação
Linguagem de programação I
Prof.: Francisco Glaubos

6,7

Prova 2 - Tipo 2

2023.2

1) (2,0 pts) Qual será a saída dos seguintes programa em C?

a)
```
#include <stdio.h>
int main() {
  int *pc = NULL, a = 3, c = 5;
  pc = &a;  *pc = c;   c = 1;
  printf("%d %d", a, *pc);
  return 0;
}
```

I. 5 1 II. 1 5 III. 5 5 IV. 1 3 V. 3 5

b)
```
#include <stdio.h>
typedef union {
  float salary;
  int workerNo;
} Job;

int main() {
  Job j;
  j.salary = 1.3;  j.workerNo = 5;
  printf("%.1f %d", j.salary,
j.workerNo);
  return 0;
}
```

I. 5 0.0 II. 1.3 5 III. 0 1.3 IV. 0.0 5

2) (8,0 pts) Você está coordenando a distribuição de recursos em um evento comunitário. Cada participante tem uma necessidade mínima (`necessidades[i]`) e você dispõe de recursos com valores variados (`recursos[j]`). Se um recurso `j` atende ou supera a necessidade de um participante `i`, ele pode ser alocado para este participante, satisfazendo sua necessidade.

Seu objetivo é alocar os recursos para maximizar o número de participantes satisfeitos. Além disso, você deve registrar em um arquivo quais recursos foram alocados para quais participantes.

Você deve implementar a seguinte função[1]:

```
void alocar_recursos(struct
DistribuicaoRecursos dados, const
char* nomeArquivo);
```

O arquivo deve conter:

A cada linha, um participante e o recurso alocado para ele, no formato `"Participante i recebeu o recurso j"`. No final, deve ser mostrado o número de participantes satisfeitos.

Exemplo 1:

Entrada:
```
int necessidades[] = {1, 2, 3};
int recursos[] = {1, 1};
DistribuicaoRecursos dados =
{necessidades, recursos, 3, 2};
```

Arquivo de Saída:
```
Participante 0 recebeu recurso 1
1
```

Exemplo 2:

Entrada:
```
int necessidades[] = {1, 2};
int recursos[] = {1, 2, 3};
DistribuicaoRecursos dados =
{necessidades, recursos, 3, 2};
```
2 3

Arquivo de Saída:
```
Participante 0 recebeu recurso 1
Participante 1 recebeu recurso 2
2
```

[1] É proibido o uso de variáveis globais

1) a. III) 5 5 ✓

b. IV) 0,0 5 ✓

2,0

2)
```
typedef struct{
    int necessidades[50];
    int recursos[50];
    int participantes;
    int qtd_recursos;
} distribuição_recursos;
```
ponteiros ✗

```
void alocar_recursos(struct distribuição_recursos dados; const char *nome_arq
    FILE *F1 = Fopen( nomearquivo, "W");
    int selecionados [dados.participantes];
    int entregue [dados.qtd_recursos];
    int satisfeitos = 0;

    for(int i = 0; i < dados.participantes; i++){
        for(int j = 0; j < dados.qtd_recursos; j++){
            ordenar(dados.necessidades);
            if(dados[j].recursos >= dados[i].necessidades){
                selecionados = i;
                entregue = dados[j].recursos;
                satisfeitos++;
            }
        }
    }
    for(int i = 0; i < sizeof(selecionados); i++){
        Fprintf(F1, "participante %d recebe recurso %d \n", selecionados[i],
                                                          entregue[i]);
    }
    Fputs(satisfeitos);
    Fclose(F1);
}
```

✗ → se menciona deve implementar

não é um array? ✗

↘ não entendi

→ dessa forma irá para o arquivo apenas o último

// a função ordenar ordena o vetor necessidades, enquanto a função alocar_recursos itera sobre os índices de necessidades e recursos para checar se o valor de recursos é maior ou igual ao de necessidade. a função também abre um arquivo, coloca as informações solicitadas nele, e então o fecha encerrando o programa.

resposta: 29 / 11 / 23

2) continuação da segunda questão.

```
Void ordenar(int vetor){
    int aux;
    for(int i=0; i<tamanho(vetor); i++){
        for(int j = tamanho(vetor)-1; j>-1; j--){
            if(vetor[i] < vetor[j]){
                aux = vetor[i];
                vetor[i] = vetor[j];
                vetor[j] = aux;
            }
        }
    }
return vetor;
```