UNIVERSIDADE FEEVALE

INSTITUTO DE CIÊNCIAS CRIATIVAS E TECNOLÓGICAS

NOME Fábio Henrique Araújo

DISCIPLINA ESTRUTURA DE DADOS / ESTRUTURA DE DADOS I

CURSO Ciência da Computação

PROFESSOR JULIANO VARELLA DE CARVALHO DATA 16 / 06 / 2022

Segunda Avaliação: parte teórica

[2,0] 1) Observe o algoritmo de ordenação *sort1*, detalhado abaixo. Considere que o *vetor A* recebe inicialmente os seguintes elementos {5,3,4,6,1,2} e a constante TAMANHO tem valor igual a 6. A partir destas observações, responda as 2 perguntas abaixo.

SORT1

```
1   void sort1(int A[]) {
2        int i, j, k, menor, aux;
3        for (i=0; i< TAMANHO; i++) {
4             menor = i;
5             for (j=i+1; j< TAMANHO; j++)
6               if (A[j] < A[menor])
7                  menor = j;
8             aux = A[menor];
9             A[menor] = A[i];
10            A[i] = aux;
11
12            for (k=0; k<TAMANHO; k++)
13              printf("%d ", A[k]);
14            printf("\n");
15       }
16  }
```

a) Quais as impressões que ocorrem na linha 13 do algoritmo *sort1*?

A linha 13 (dentro do for da linha 12), imprime o vetor a cada iteração do algoritmo:
1 3 4 6 5 2
1 2 4 6 5 3
1 2 3 6 5 4
1 2 3 4 5 6
1 2 3 4 5 6
1 2 3 4 5 6

b) Explique a estratégia de ordenação utilizada por esse algoritmo e diga qual o nome dele.

Nesse algoritmo foi utilizado a estratégia de ordenação conhecida como Selection Sort, que consiste em ler um array a partir do primeiro número, onde se assume o número na posição [0] é o menor número e sua posição é salva, depois o algoritmo percorre o restante do array comparando o número da posição menor dentro array com o número da posição j, caso o número da posição j seja menor que o número da posição menor, menor recebe a posição de j e assim sucessivamente até o fim do array. Após ter lido e comparado todo o array com o valor definido como menor, o valor na posição **menor** é armazenado numa variável auxiliar, o valor da posição **i**é salvo na posição **menor** e a posição **i** do array recebe o valor da variável auxiliar.

[1,0] 2) Qual o valor de retorno da função a seguir, caso n = 27?

```
int recursao (int n) {
    if (n <= 10) {
        return n * 2;
    }
    else {
        return recursao(recursao(n/3));
    }
}
```

*A função retorna o valor 16, pois primeiro é passado n=27, que não é <=10, então é chamado recursao(recursao(n/3)), assim sendo, na fila é executado recursao(9), devido ter sido feito n/3, 9 é <=10, então a função retorna 18, aí é executado a segunda chamada de recursao com 18 – recursao(18), como 18 não é <=10 é executado recursao(recursao(18/3)), que executará recursao(6), onde n é <=10 e retornará n*2, que é 12, será executada uma nova recursão de 12, que não é <=10 e chamará novamente recursao(recursao(n/3)), que executa recursao(4), que é <=10 e retornará n*2, que dá 8, ocorrerá uma nova recursão em cima do valor 8 – recursao(8), que é <=10, desempilhando a últiam execução de recursao e retornando por fim o número 16.*

[1,0] 3) O algoritmo *Bubble Sort* é bastante popular. Usando-se esse algoritmo, sem qualquer otimização, para ordenar um vetor alocado sequencialmente, em ordem crescente, contendo os números [5, 4, 1, 3, 2], serão feitas:

Resposta: D.

a) 10 comparações e 7 trocas

b) 10 comparações e 10 trocas

c) 16 comparações e 9 trocas

d) 10 comparações e 8 trocas

e) 16 comparações e 10 trocas

[1,0] 4) De acordo com a função recursiva abaixo, implementada usando a linguagem C.

```
int prova (int N) {
   if (N == 0) return 0;
   else return N * 2 - 1 + prova(N - 1);
}
```

Considerando-se que essa função sempre será chamada com a variável N contendo inteiros positivos, o seu valor de retorno será:

Resposta: A.

a) O valor armazenado em N elevado ao quadrado.

b) O fatorial do valor armazenado em N.

c) O somatório dos N primeiros números inteiros positivos.

d) O somatório dos N primeiros números pares positivos.

e) 2 elevado ao valor armazenado em N.

[1,0] 5) Observe o código abaixo e responda as questões de (a) à (c).

```
1   #include <stdio.h>
2   #define TAM 5
3
4   void funcao1(int v[], int l, int r) {
5       int a;
6       if (l >= r) return;
7       else {
8           a = v[l];
9           v[l] = v[r];
10          v[r] = a;
11          funcao1(v, l+1, r-1);
12      }
13  }
14
15  int main(int argc, const char * argv[]) {
16      int v[11] = {1,2,3,4,5};
17      funcao1(v, 0, TAM-1);
18      for (int i = 0; i < TAM; i++)
19          printf("%d\t", v[i]);
20      return 0;
21  }
```

(a) O que será impresso na linha 19 ?

A função funcao1 inverte o array, e posteriormente a linha 19 (dentro do for da linha 18), imprime os valores de **v[i]** e adiciona um tab em seguida.
Portando, é impresso:
5 4 3 2 1

(b) Mostre, com detalhes, a pilha de execução após a chamada da *funcao1*.

Ao chamar a funcao1, será empilhado a função 1 vez, que irá trocar os valores 1 e 5 de lugar e entrará na recursão, chamando novamente a funcao1, tendo assim 2 na pilha, como ocorre novamente a troca dos valores (neste caso 2 e 4) é chamado novamente a funcao1, como l é 2 e r também é 2, ela é executada direto, e retorna desempilhando o restante.
Assim, após desempilhar as chamadas da funcao1, permanece na pilha de execução somente o main(int argc, const char * argy[]).

Segunda Avaliação: parte prática

[4,0] **1) [O programa deve ser implementado por completo, com todas as funções necessárias para o seu funcionamento]**

Um palíndromo é uma palavra que, quando soletrada do início para o final e do final para o início, soa da mesma maneira. Por exemplo, a palavra Arara é um palíndromo, assim como reviver, anilina, ovo, dentre outras.

a) Desenvolva um algoritmo **recursivo** para descobrir se uma palavra é um palíndromo, lembrando que:

Caso a palavra seja vazia ou contenha somente uma letra, ela é um palíndromo.

Caso a palavra tenha mais de um caractere, deve-se comparar a primeira letra com a última letra da palavra.

Se a primeira e a última letra são distintas, então a palavra não é um palíndromo. Caso contrário, a segunda letra deve ser comparada com a penúltima letra, e assim sucessivamente.

b) Desenvolva um algoritmo **iterativo** para descobrir se uma palavra é um palíndromo.

Boa avaliação