1)

Programação funcional é um paradigma orientado ao uso de funções, em que essas funções recebem argumentos e dão um retorno ao usuário. Linguagens de programação que seguem esse paradigma são chamadas de linguagens funcionais. Como por exemplo Haskell e Scala.

As vantagens do uso desse paradigma seria a otimização e capacidade de reutilização de algoritmos através do uso de funções e o paralelismo e concorrência de códigos. Porém a performace das linguagens funcioansi são inferiores às demais, além do fato da dificuldade de prever os quisitos de um sistema para sua implementação, quando comparado com outras linguagens.

2)

1 - Funções podem retornar funções ou receber funções como argumento.

2 - Variáveis são imutáveis ao decorrer do algoritmo.

3 - À partir de uma função existente é possível gerar outra função que as execute simultaneamente.

```
fact :: Int -> Int

fact 0 = 1

fact n = n * fact (n-1)


main = print $ fact 7
```

4)

Em programação orientada a objeto um problema é estudado, conceituado, e com isso aplicado o modelo de orientação a objeto. Estes modelos (" objetos ") são dadas funcionalidade ("métodos") que lhes permitem realizar várias ações , e cada objeto também pode armazenar informações. A unidade fundamental em OOP é o objeto , o que em si pode conter outros objetos, um pouco como uma boneca Matryoshka do russo . A idéia é tornar mais fácil para programadores humanos para visualizar o programa , amarrando o programa em concreto coisas , que permite agrupar as coisas logicamente .

5)

int year; visível em toda a calsse

String make; visível em toda a classe

double speed; visível em toda classe

public Car(int y, string m, double beginningSpeed)

\/

parâmetros - escopo local

int tmp = year; variável local do tipo primitivo

Roda r = new Roda (tmp); variável local (objeto)

6)

Ao executar o código ocorrerá um erro relacionado à referência de memória devido ao fato do valor da variável (b) ser um endereço de memória alocado dinamicamente. Ao realizar a manipulação deste endereço (linha 21) ocorre uma atribuição juntamente com o acréscimo desta variável, que é um endereço de memória. Com isso gera o erro de Runtime Error.

7 - A)

Haskell

```
main = do
inputjar <- getLine
let n1 = read inputjar::Double
inputjar <- getLine
let n2 = read inputjar::Double
inputjar <- getLine
let n3 = read inputjar::Double
```

```
        inputjar <- getLine
        let n4 = read inputjar::Double
        let media = (n1+n2+n3+n4)/4
        if media>=7 then do
                print("Aprovado")
                print(media)
        else do
                inputjar <- getLine
                let ex = read inputjar::Double
                let fin = (media+ex)/2
                if fin>=5 then do
                        print("Aprovado em Exame")
                        print(fin)
                else do
                        print("Reprovado")
                        print(fin)
```

Java

```
 import java.util.Scanner;
import java.util.*;
import java.lang.*;
import java.io.*;

class Ideone{
        public static void main (String[] args){
                double media,ex,fin;
```

```
            double n1,n2,n3,n4;
            Scanner ler=new Scanner(System.in);
            System.out.println("Digite as 4 notas do estudante");
            n1=ler.nextDouble();
            n2=ler.nextDouble();
            n3=ler.nextDouble();
            n4=ler.nextDouble();
            media=(n1+n2+n3+n4)/4;
            if(media>=7){
                System.out.println("Aluno Aprovado");
                System.out.println(media);
            } else {
                ex=ler.nextDouble();
                fin=(ex+media)/2;
                if(fin>=5){
                    System.out.println("Aluno Aprovado em Exame");
                    System.out.println(fin);
                } else {
                    System.out.println("Aluno Reprovado");
                    System.out.println(fin);
                }
            }
        }
}
```

B)

Haskell

```
main = do
  inputjar <- getLine
  let n1 = read inputjar::Int
  if n1 `mod` 2 == 0 then print("número par")
  else print("número impar")
```

Java

```
import java.util.Scanner;
import java.util.*;
import java.lang.*;
import java.io.*;

class Ideone{
	public static void main (String[] args){
		int n1;
		Scanner ler=new Scanner(System.in);
		n1=ler.nextInt();
		if(n1%2==0){
			System.out.println("número par");
		} else {
			System.out.println("número impar");
		}
	}
}
```

C)

Java

```
import java.util.Scanner;
import java.util.*;
import java.lang.*;
import java.io.*;

class Ideone{
	public static void main (String[] args){
		int tam,i,j;
		Scanner ler=new Scanner(System.in);
		System.out.println("Digite o tamanho do vetor:");
		tam=ler.nextInt();
		int A[]=new int[tam];
		System.out.println("Digite os valores a serem utilizados no vetor");
		for(i=0;i<tam;i++){
			A[i]=ler.nextInt();
		}
		for(i=0;i<tam;i++){
			j=i;
			while(j>0 && A[j-1]>A[j]){
				A[j-1]^=A[j];
				A[j]^=A[j-1];
				A[j-1]^=A[j];
				j=j-1;
			}
```

```
                }
                System.out.println("vetor ordenado");
                for(i=0;i<tam;i++){
                        System.out.printf("%d ",A[i]);
                }
        }
}
```

Haskell

```
srt::(Ord a)=>[a]->[a]
srt[]=[]
srt lista=srt1 lista(length lista)

srt1::(Ord a)=>[a]->Int->[a]
srt1 lista 0=lista
srt1 lista n=srt1(swp lista)(n-1)

swp::(Ord a)=>[a]->[a]
swp[x]=[x]
swp(x:y:zs)
        |x>y=y:swp(x:zs)
    |otherwise=x:swp(y:zs)

main = do
        let k=[1,7,6,8,40,-45,3]
        let m=srt(k)
        print("vetor")
```

```
        print(k)
        print("vetor ordenado")
        print(m)
```

D)

Java

```
import java.util.Scanner;
import java.util.*;
import java.lang.*;
import java.io.*;

class Ideone{
        public static void main (String[] args){
                Scanner ler=new Scanner(System.in);
                int tam,i,j,min,max;
                System.out.println("Digite o tamanho do vetor");
                tam=ler.nextInt();
                int vet[]=new int[tam];
                System.out.println("Digite os valores do vetor");
                for(i=0;i<tam;i++){
                        vet[i]=ler.nextInt();
                }
                min=vet[0];
                max=vet[0];
                for(i=1;i<tam;i++){
                        if(vet[i]<min){
```

```
                    min=vet[i];
                }
                if(vet[i]>max){
                    max=vet[i];
                }
            }
            System.out.println("maior valor:");
            System.out.println(max);
            System.out.println("menor valor:");
            System.out.println(min);
        }
}
```

Haskell

```
maxi [x] = x
maxi (x:xs)
    | x > maxi xs = x
    | otherwise = maxi xs
mini [x] = x
mini (x:xs)
        | x < mini xs = x
        | otherwise = mini xs
main = do
        let k = [1,7,6,8,40,-45,3]
        let m = maxi k
        let n = mini k
        print("maior:")
```

```
print(m)
print("menor:")
print(n)
```