***Assignment#4:***

* Q1: (using variables)

#[derive(Debug)]

enum TypesOfStudents {

Boys {present:bool},

Girls {present:bool},

}

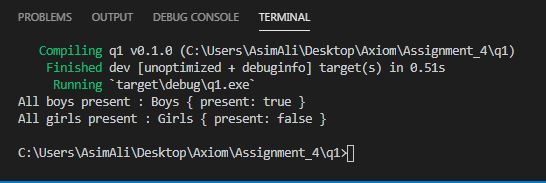
fn main() {

let stu1=TypesOfStudents::Boys{present:true};

let stu2=TypesOfStudents::Girls{present:false};

println!("All boys present : {:?}\nAll girls present : {:?}",stu1,stu2 )

}



* Q1.2: (using function)

#[derive(Debug)]

enum TypesOfStudents {

Boys {present:bool},

Girls {present:bool},

}

fn stu1() -> TypesOfStudents {

TypesOfStudents::Boys{present:true}

}

fn stu2() -> TypesOfStudents {

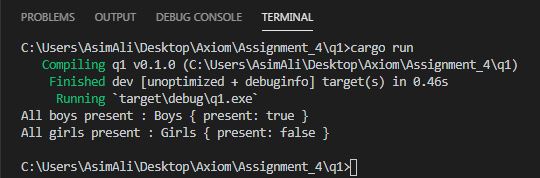
TypesOfStudents::Girls{present:false}

}

fn main() {

println!("All boys present : {:?}\nAll girls present : {:?}",stu1(),stu2() )

}



* Q2:

#[derive(Debug)]

enum TypesOfStudents {

Boys (String,u32,f32),

Girls (String,u32,f32),}

fn main() {

let stu1=TypesOfStudents::Boys("Ali".to\_string(),22,5.9);

let stu2=TypesOfStudents::Girls("Ayesha".to\_string(),20,5.4);

println!("Boys(Name,Age,Height) : {:?}",stu1 );

println!("Girls(Name,Age,Height) : {:?}\n",stu2 );

if let TypesOfStudents::Boys(name,age,height)= stu1 {

println!("Name of boy is : {:?}",name )

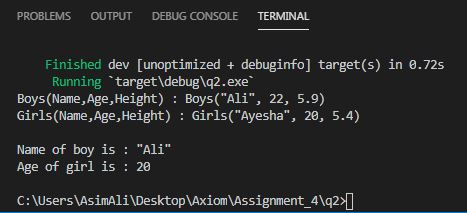
}

if let TypesOfStudents::Girls(name,age,height)= stu2 {

println!("Age of girl is : {:?}",age )

}

}



* Q3:

#[derive(Debug)]

enum TypeOfStudent {

Boys(StudentDetails),

Girls(StudentDetails),

}

#[derive(Debug)]

struct StudentDetails {

name: String,

age: u32,

height: f32

}

fn stu1() -> TypeOfStudent {

TypeOfStudent::Boys(StudentDetails{name:"Ali".to\_string(),age:22,height:5.9})

}

fn stu2() -> TypeOfStudent {

TypeOfStudent::Girls(StudentDetails{name:"Ayesha".to\_string(),age:20,height:5.4})

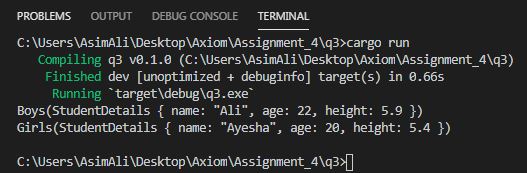
}

fn main() {

println!("{:?}",stu1());

println!("{:?}",stu2());

}



* Q4

use std::io;

#[derive(Debug)]

enum Currency {

Dollar(String),

Yen(String),

Pound(String),

Other(String)

}

fn curr(key: &str) -> Currency {

if key=="Dollar"{

let converter = Currency::Dollar("dollar".to\_string());

converter

}

else if key=="Yen"{

let converter = Currency::Yen("yen".to\_string());

converter

}

else if key=="Pound"{

let converter = Currency::Pound("pound".to\_string());

converter

}

else {

let converter = Currency::Other("other".to\_string());

converter

}

}

fn main() {

println!("Please input a Currency name only following were available:\n'Dollar', 'Yen' or 'Pound'");

let mut key = String::new();

io::stdin().read\_line(&mut key)

.expect("Failed to read text");

let key:&str=key.trim();

let converter = curr(&key);

loop {

println!("\nPlease enter your value in {} to convert it into Rs 'PKR'",key);

let mut value = String::new();

io::stdin().read\_line(&mut value)

.expect("Failed to read line");

let value: f32 = match value.trim().parse() {

Ok(num)=>num,

Err(\_)=>continue

};

match converter {

Currency::Dollar(\_dollar) => println!("\n{} {} equals Rs{} PKR", value, key, value\*155.26),

Currency::Yen(\_yen) => println!("\n{} {} equals Rs{} PKR", value, key, value\*1.42),

Currency::Pound(\_pound) => println!("\n{} {} equals Rs{} PKR", value, key, value\*200.45),

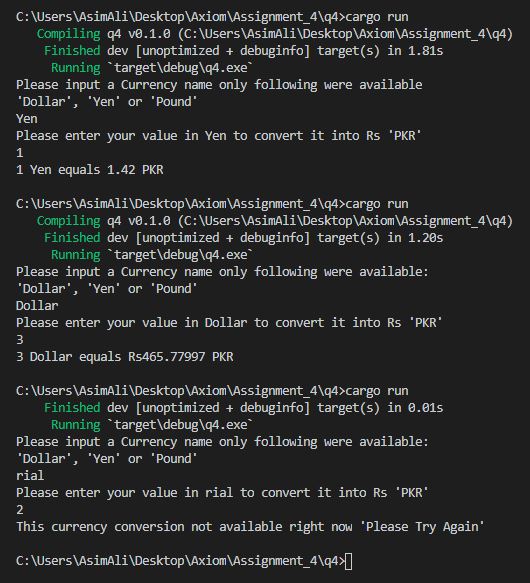
\_ => println!("\nThis currency conversion not available right now 'Please Try Again'"),

}

break

}

}



* Q5:

use std::io;

#[derive(Debug)]

enum Currency {

Dollar(String),

Yen(String),

Pound(String),

Other(String)

}

fn curr(key: &str) -> Currency {

if key=="Dollar"{

let converter = Currency::Dollar("dollar".to\_string());

converter

}

else if key=="Yen"{

let converter = Currency::Yen("yen".to\_string());

converter

}

else if key=="Pound"{

let converter = Currency::Pound("pound".to\_string());

converter

}

else {

let converter = Currency::Other("other".to\_string());

converter

}

}

fn main() {

println!("Please input a Currency name only following were available:\n'Dollar', 'Yen' or 'Pound'");

let mut key = String::new();

io::stdin().read\_line(&mut key)

.expect("Failed to read text");

let key:&str=key.trim();

let converter = curr(&key);

loop {

println!("\nPlease enter your value in {} to convert it into Rs 'PKR'",key);

let mut value = String::new();

io::stdin().read\_line(&mut value)

.expect("Failed to read line");

let value: f32 = match value.trim().parse() {

Ok(num)=>num,

Err(\_)=>continue

};

if let Currency::Dollar(\_dollar) = &converter {

println!("\n{} {} equals {} PKR",value, key, value\*155.62 )

}

if let Currency::Pound(\_pound) = &converter {

println!("\n{} {} equals {} PKR",value, key, value\*1.42 )

}

if let Currency::Yen(\_yen) = &converter {

println!("\n{} {} equals {} PKR",value, key, value\*200.45 )

}

if let Currency::Other(\_other) = &converter {

println!("\nThis currency conversion not available right now 'Please Try Again'")

}

break

}

}

