**Internship Test**

Rust Developer Profile Set-1

Name – PRUSHOTTAM VERMA

BRANCH – COMPUTER SCIENCE AND BUSINESS SYSTEM

Question 1- Implement a function that checks whether a given string is a palindrome or not.

Solution-

use std::io;

fn main() {

println!("Enter a string:");

let mut input = String::new();

io::stdin().read\_line(&mut input).expect("Failed to read line");

let input = input.trim();

if is\_palindrome(input) {

println!("The string \"{}\" is a palindrome.", input);

} else {

println!("The string \"{}\" is not a palindrome.", input);

}

}

fn is\_palindrome(s: &str) -> bool {

let mut left = 0;

let mut right = s.len() - 1;

while left < right {

if s.chars().nth(left) != s.chars().nth(right) {

return false;

}

left += 1;

right -= 1;

}

true

}

Question 2- Given a sorted array of integer, implement a function that returns the index of the first occurrence of a given number.

Solution-

fn main() {

let a = vec![1, 2, 3, 4, 4, 4, 5, 6, 7, 8];

let t = 4;

match find\_first\_occurrence(&a, t) {

Some(i) => println!("The first occurrence of {} is at index {}", t, i),

None => println!("The number {} is not found in the array", t),

}

}

fn find\_first\_occurrence(arr: &[i32], target: i32) -> Option<usize> {

let mut l = 0;

let mut r = arr.len();

while l < r {

let m = l + (r - l) / 2;

if arr[m] < target {

l = m + 1;

} else {

r = m;

}

}

if l < arr.len() && arr[l] == target {

Some(l)

} else {

None

}

}

Question 3- Given a string of word, implement a function that returns the shortest word in the string.

Solution –

fn main() {

let sentence = "The quick brown fox jumps over the lazy dog";

match find\_shortest\_word(sentence) {

Some(shortest) => println!("The shortest word is: {}", shortest),

None => println!("The string is empty"),

}

}

fn find\_shortest\_word(sentence: &str) -> Option<&str> {

sentence

.split\_whitespace()

.min\_by\_key(|word| word.len())

}

Question 4- Implement a function that check whether a given number is prime or not.

Solution –

fn main() {

let n = 17;

if is\_prime(n) {

println!("{} is a prime number", n);

} else {

println!("{} is not a prime number", n);

}

}

fn is\_prime(num: u64) -> bool {

if num <= 1 {

return false;

}

let mut i = 2;

while i \* i <= num {

if num % i == 0 {

return false;

}

i += 1;

}

true

}

Question 5- Given a sorted array of integers, implement a function that return the median of the array.

Solution-

fn main() {

let arr = vec![1, 2, 3, 4, 5, 6, 7, 8, 9];

let median\_value = find\_median(&arr);

println!("The median of the array is: {}", median\_value);

}

fn find\_median(nums: &[i32]) -> f64 {

let len = nums.len();

if len % 2 == 0 {

let mid\_right = len / 2;

let mid\_left = mid\_right - 1;

(nums[mid\_left] + nums[mid\_right]) as f64 / 2.0

} else {

nums[len / 2] as f64

}

}

Question 6- Implement a function that finds the longest common prefix of a given set of strings.

Solution –

fn main() {

let s = vec![

String::from("flower"),

String::from("flow"),

String::from("flight"),

];

let c = find\_longest\_common\_prefix(&s);

match c {

Some(p) => println!("The longest common prefix is: {}", p),

None => println!("There is no common prefix"),

}

}

fn find\_longest\_common\_prefix(strs: &[String]) -> Option<String> {

if strs.is\_empty() {

return None;

}

let f = &strs[0];

let mut p = String::new();

for (i, c) in f.chars().enumerate() {

for s in strs.iter().skip(1) {

if let Some(ch) = s.chars().nth(i) {

if ch != c {

return Some(p);

}

} else {

return Some(p);

}

}

p.push(c);

}

Some(p)

}

Question 7- Implement a function that returns the kth smallest elements in a given array.

Solution –

fn main() {

let arr = vec![7, 10, 4, 3, 20, 15];

let k = 3;

let kth\_smallest = find\_kth\_smallest(&arr, k);

println!("The {}-th smallest element is: {}", k, kth\_smallest);

}

fn find\_kth\_smallest(nums: &[i32], k: usize) -> i32 {

let mut sorted\_nums = nums.to\_vec();

sorted\_nums.sort();

sorted\_nums[k - 1]

}

Question 8- Given a binary tree, implement a function that return the maximum depth of the tree.

Solution –

#[derive(Debug)]

struct TreeNode {

val: i32,

left: Option<Box<TreeNode>>,

right: Option<Box<TreeNode>>,

}

impl TreeNode {

fn new(val: i32) -> Self {

TreeNode { val, left: None, right: None }

}

}

fn main() {

// Create a sample binary tree

let root = Some(Box::new(TreeNode {

val: 3,

left: Some(Box::new(TreeNode::new(9))),

right: Some(Box::new(TreeNode {

val: 20,

left: Some(Box::new(TreeNode::new(15))),

right: Some(Box::new(TreeNode::new(7))),

})),

}));

let depth = max\_depth(root);

println!("The maximum depth of the binary tree is: {}", depth);

}

fn max\_depth(root: Option<Box<TreeNode>>) -> i32 {

match root {

Some(node) => {

let left\_depth = max\_depth(node.left);

let right\_depth = max\_depth(node.right);

1 + left\_depth.max(right\_depth)

}

None => 0,

}

}

Question 9 – Reverse a string in rust .

Solution –

fn main() {

let original\_string = "hello";

let reversed\_string = reverse\_string(original\_string);

println!("Original String: {}", original\_string);

println!("Reversed String: {}", reversed\_string);

}

fn reverse\_string(s: &str) -> String {

s.chars().rev().collect()

}

Question 10- Check if a number is prime in Rust.

Solution –

fn main() {

let num = 17;

if is\_prime(num) {

println!("{} is a prime number", num);

} else {

println!("{} is not a prime number", num);

}

}

fn is\_prime(num: u64) -> bool {

if num <= 1 {

return false;

}

let mut divisor = 2;

while divisor \* divisor <= num {

if num % divisor == 0 {

return false;

}

divisor += 1;

}

true

}

Question 11- Merge two sorted arrays in rust.

Solution –

fn main() {

let arr1 = vec![1, 3, 5, 7, 9];

let arr2 = vec![2, 4, 6, 8, 10];

let merged\_array = merge\_sorted\_arrays(&arr1, &arr2);

println!("Merged Array: {:?}", merged\_array);

}

fn merge\_sorted\_arrays(arr1: &[i32], arr2: &[i32]) -> Vec<i32> {

let mut merged = Vec::with\_capacity(arr1.len() + arr2.len());

let (mut i, mut j) = (0, 0);

while i < arr1.len() && j < arr2.len() {

if arr1[i] < arr2[j] {

merged.push(arr1[i]);

i += 1;

} else {

merged.push(arr2[j]);

j += 1;

}

}

merged.extend\_from\_slice(&arr1[i..]);

merged.extend\_from\_slice(&arr2[j..]);

merged

}

Question 12- Find the maximum subarray sum in rust.

Solution –

fn main() {

let arr = vec![-2, 1, -3, 4, -1, 2, 1, -5, 4];

let max\_sum = max\_subarray\_sum(&arr);

println!("Maximum subarray sum: {}", max\_sum);

}

fn max\_subarray\_sum(nums: &[i32]) -> i32 {

let mut max\_ending\_here = 0;

let mut max\_so\_far = std::i32::MIN;

for &num in nums {

max\_ending\_here = max\_ending\_here.max(0) + num;

max\_so\_far = max\_so\_far.max(max\_ending\_here);

}

max\_so\_far

}