**Reg. No: 21BCE1297 Name: Vidhi Shah Date: 07/04/22**

**PPS10**

**Q1**

**Aim:**

There are ‘n’ concentric rectangles one inside another. The length and breadth of the surrounding rectangle is one unit more than the inner one. Write a C program with a recursive function that finally returns the area of the outermost rectangle. Get the number of rectangles ‘n’ and dimensions (length and breadth) of innermost rectangle as user input.

**Procedure:**

**Input:**

Number of rectangles, ‘n’

Length of outermost rectangle, ‘l’

Breadth of outermost rectangle, ‘b’

**Output:**

Area of outermost rectangle

**Algorithm:**

Step 1: Declare ‘arearect’ function with return type ‘int’ and arguments ‘int l’, ‘int b’ and ‘int n’.

**Main Function**

Step 1: Read integer variables ‘l’, ‘b’ and ‘n’.

Step 2: Call ‘arearect’ function with input parameters ‘l’, ‘b’ and ‘n’.

Step 3: Print the area (return value of the ‘arearect’ function)

Step 4: Return 0

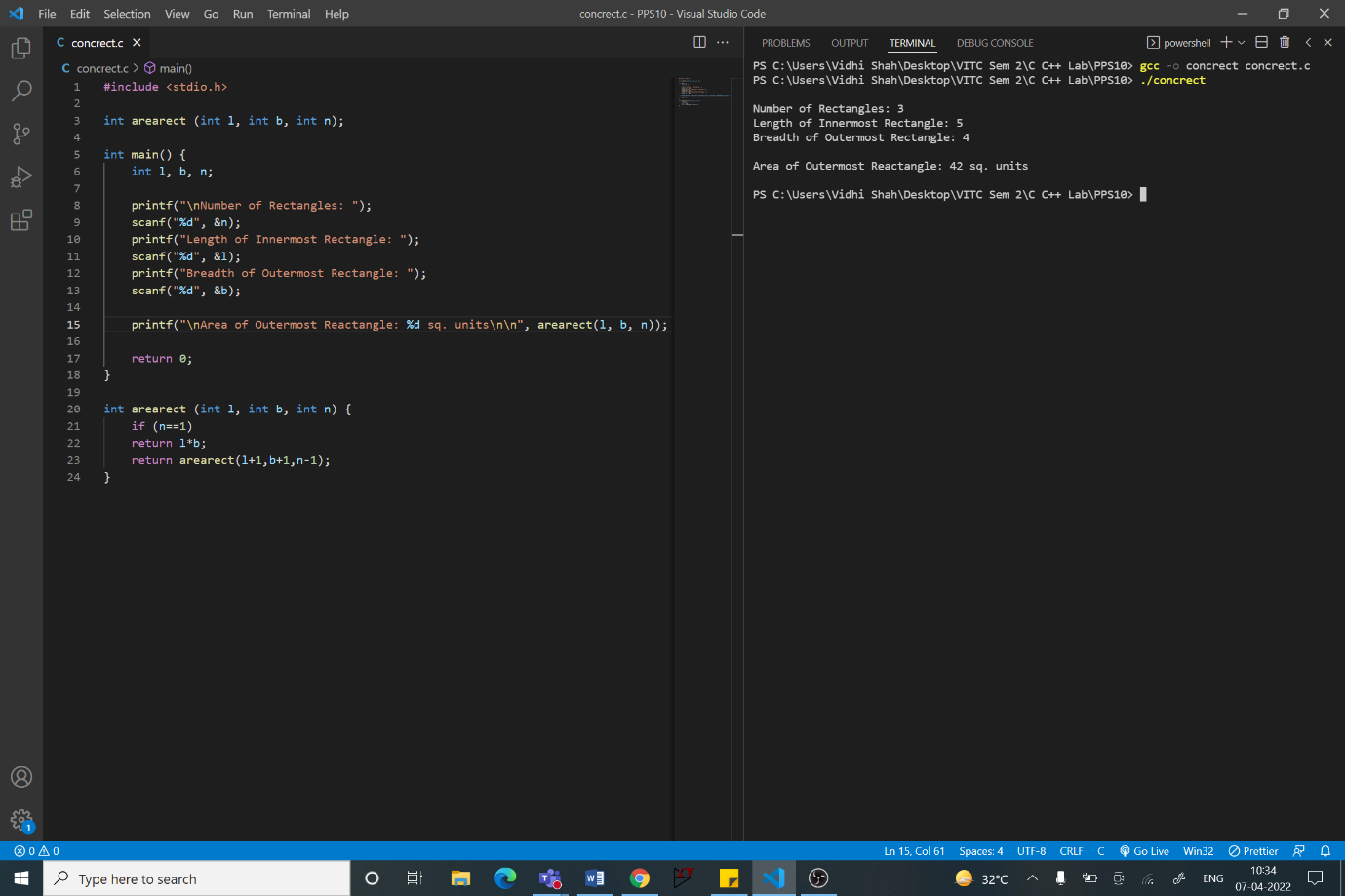
**AreaRect Function**

Step 1: If n is equal to 1

Step A: Return l\*b

Step 2: Return arearect(l+1, b+1, n-1)

**Code:**



#include <stdio.h>

int arearect (int l, int b, int n);

int main() {

    int l, b, n;

    printf("\nNumber of Rectangles: ");

    scanf("%d", &n);

    printf("Length of Innermost Rectangle: ");

    scanf("%d", &l);

    printf("Breadth of Outermost Rectangle: ");

    scanf("%d", &b);

    printf("\nArea of Outermost Reactangle: %d sq. units\n\n", arearect(l, b, n));

    return 0;

}

int arearect (int l, int b, int n) {

    if (n==1)

    return l\*b;

    return arearect(l+1,b+1,n-1);

}

**Q2**

**Aim:**

Write a ‘C’ program using function pointers to insert a number ‘n’ at position ‘p’ of an array.

**Procedure:**

**Input:**

Number of elements in the array, ‘x’

Number to be inserted, ‘n’

Index of new element, ‘p’

Elements of the array

**Output:**

Array with inserted element

**Algorithm:**

Step 1: Declare global integer variable, ‘x’

Step 2: Declare ‘insert’ function with return type integer pointer, ‘int\*’ and arguments ‘int arr[x+1]’, int ‘n’, int ‘p’ and int ‘x’.

**Main Function**

Step 1: Read integer variables ‘x’, ‘n’ and ‘p’

Step 2: Read elements of the integer array, ‘array’

Step 3: Initialise an integer pointer variable, ‘ptr’

Step 4: Initialise a function pointer, ‘insertptr’

Step 5: Assign the address of ‘insert’ function to ‘insertptr’ function pointer

Step 6: Call the ‘insert’ function and assign the return value to pointer variable, ‘ptr’

Step 7: Print the new array using pointer variable ‘ptr’

Step 8: Return 0

**Insert Function**

Step 1: For ‘i' from x to p+1

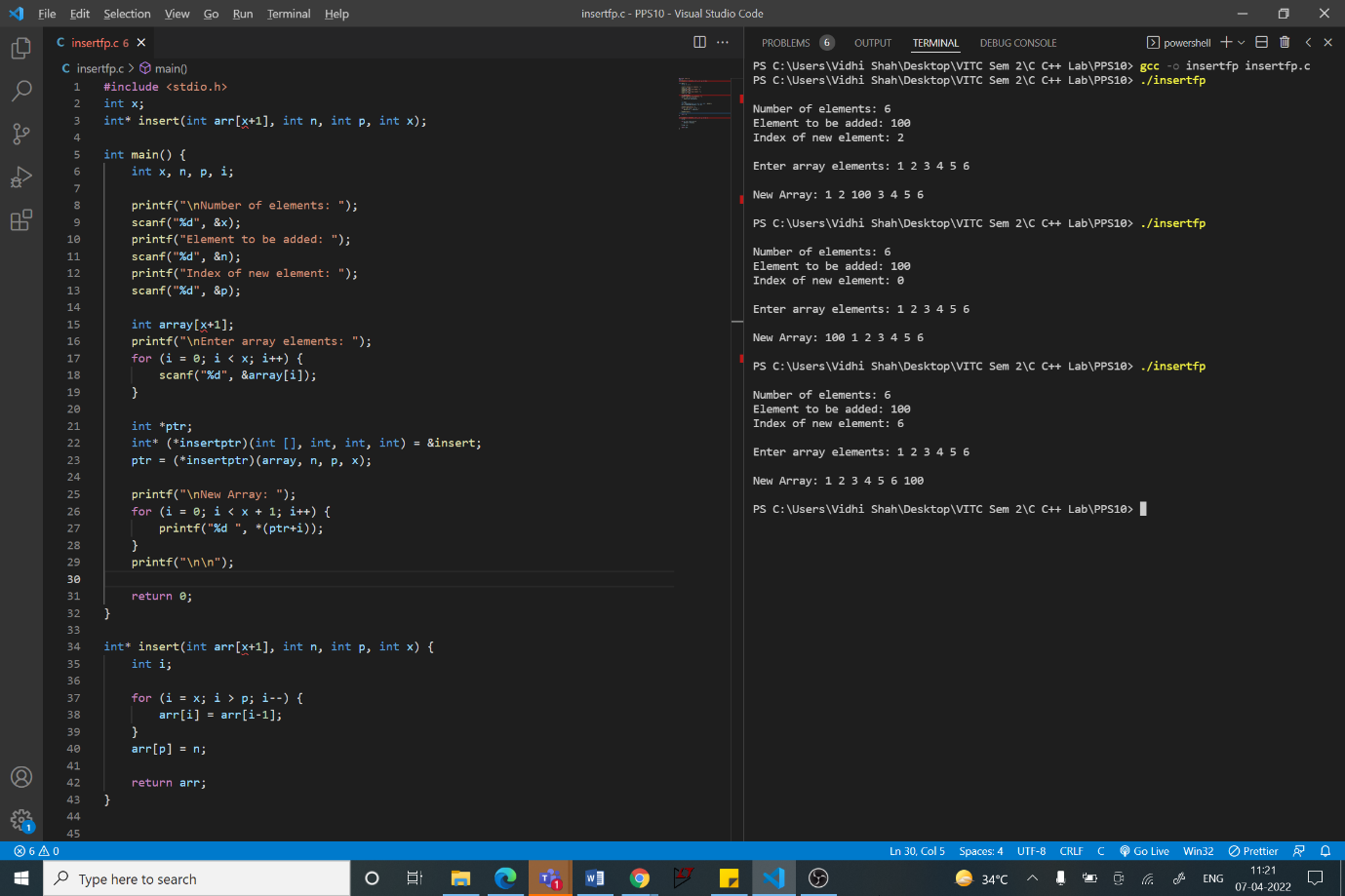
Step A: arr[i] = arr[i-1]

Step B: i = i - 1

Step 2: arr[p] = arr[n]

Step 3: Return arr (Pointer to the array)

**Code:**



**Code Snippets:**

// Function Pointer

    int \*ptr;

    int\* (\*insertptr)(int [], int, int, int) = &insert;

    ptr = (\*insertptr)(array, n, p, x);

// Pointer Arithmetic

printf("\nNew Array: ");

    for (i = 0; i < x + 1; i++) {

        printf("%d ", \*(ptr+i));

    }

// Insert function

int\* insert(int arr[x+1], int n, int p, int x) {

    int i;

    for (i = x; i > p; i--) {

        arr[i] = arr[i-1];

    }

    arr[p] = n;

    return arr;

}

**Q3**

**Aim:**

Write a ‘C’ program using function pointers to check if the given string is palindrome or not.

**Procedure:**

**Input:**

String, ‘name’

**Output:**

Given string is a Palindrome or Not a Palindrome

**Algorithm:**

Step 1: Declare ‘palindrome’ function with return type ‘int’ and arguments ‘char name[15]’

**Main Function**

Step 1: Initialise char array, ‘name’, of size 15

Step 2: Read the string from user into variable ‘name’

Step 3: Declare integer variable ‘result’

Step 4: Initialise a function pointer, ‘palindromeptr’

Step 5: Assign the address of ‘palindrome’ function to ‘palindromeptr’ function pointer

Step 6: Call the ‘palindrome’ function and assign the return value to variable ‘result’

Step 7: If result is equal to 0

Step A: Print “Palindrome”

Step 8: Else

Step B: Print “Not a Palindrome”

Step 9: Return 0

**Palindrome Function**

Step 1: Initialise char array, ‘revname’, of size 15 and integer variable ‘result’

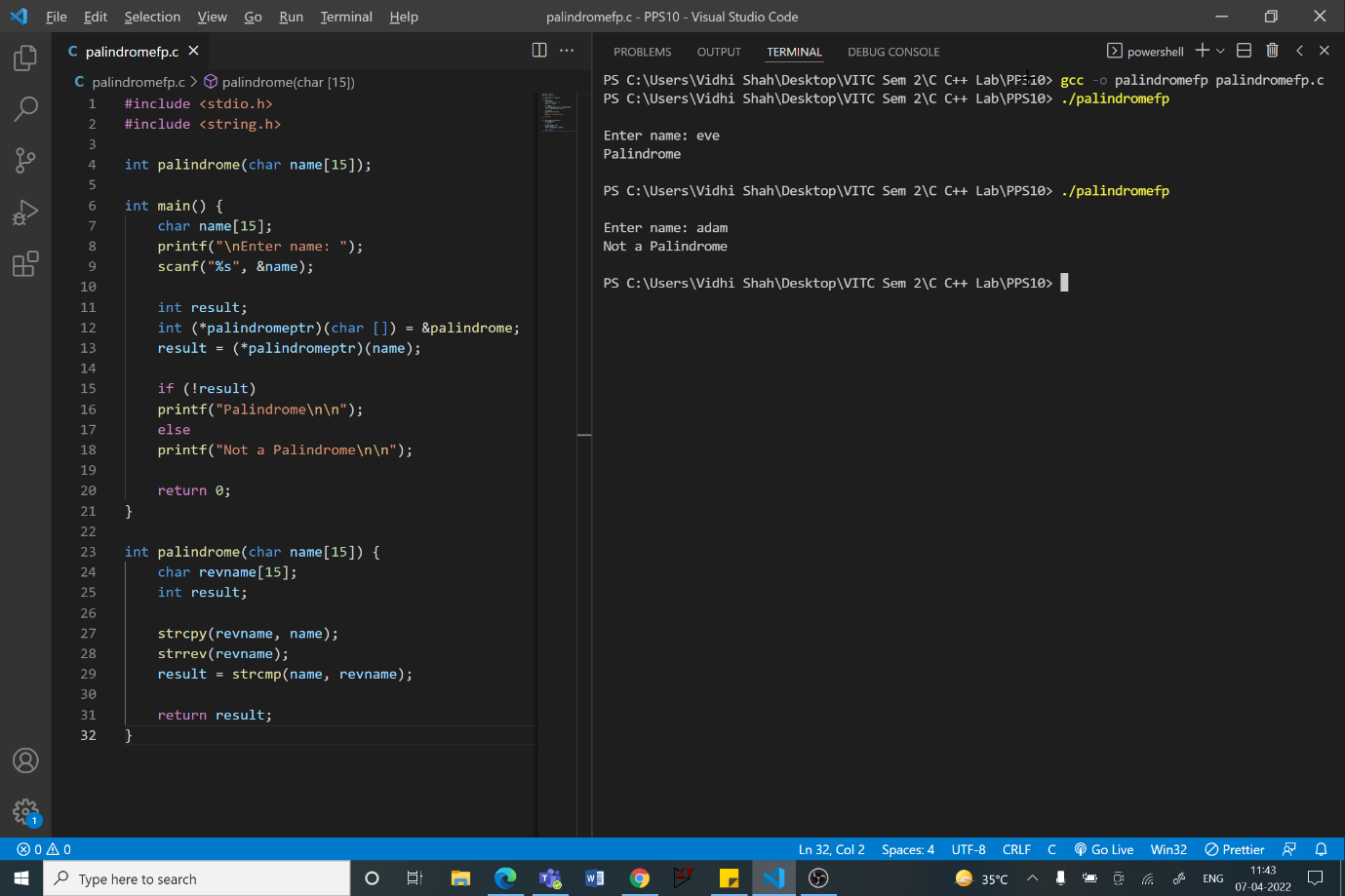
Step 2: Copy string ‘name’ to ‘revname’

Step 3: Reverse the string ‘revname’

Step 4: Compare the strings ‘name’ and ‘revname’ and store the value in ‘result’ variable

Step 5: Return ‘result’ variable

**Code:**



int palindrome(char name[15]);

int main() {

    char name[15];

    printf("\nEnter name: ");

    scanf("%s", &name);

    int result;

    int (\*palindromeptr)(char []) = &palindrome;

    result = (\*palindromeptr)(name);

    if (!result)

    printf("Palindrome\n\n");

    else

    printf("Not a Palindrome\n\n");

    return 0;

}

int palindrome(char name[15]) {

    char revname[15]; int result;

    strcpy(revname, name);

    strrev(revname);

    result = strcmp(name, revname);

    return result;}