

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

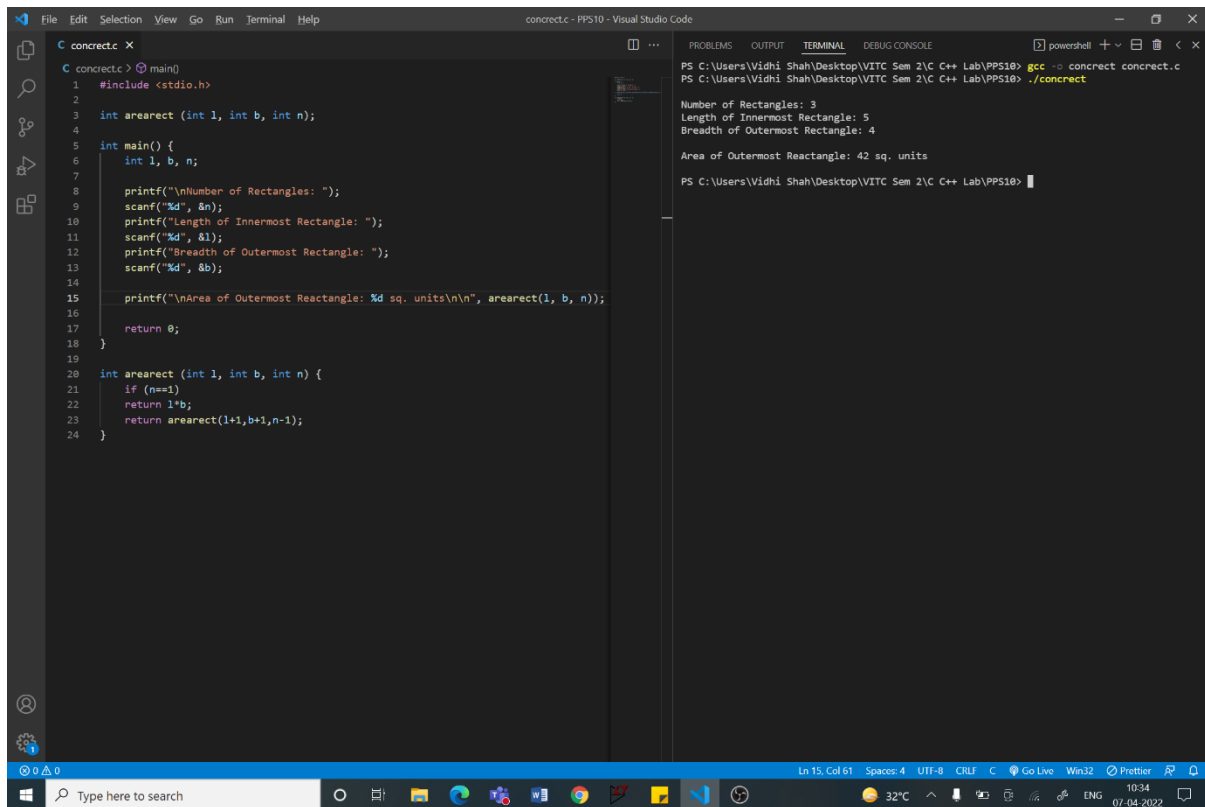
AreaRect Function

Step 1: If n is equal to 1

Step A: Return l*b

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

Code:



The screenshot shows the Visual Studio Code editor with a C++ file named `concrect.c`. The code defines a recursive function `arearect` to calculate the area of a rectangle formed by nested rectangles. The `main` function prompts the user for the number of rectangles (`n`), the length of the innermost rectangle (`l`), and the breadth of the outermost rectangle (`b`), and then prints the total area.

```
1 #include <stdio.h>
2
3 int arearect (int l, int b, int n);
4
5 int main() {
6     int l, b, n;
7
8     printf("\nNumber of Rectangles: ");
9     scanf("%d", &n);
10    printf("Length of Innermost Rectangle: ");
11    scanf("%d", &l);
12    printf("Breadth of Outermost Rectangle: ");
13    scanf("%d", &b);
14
15    printf("\nArea of Outermost Rectangle: %d sq. units\n\n", arearect(l, b, n));
16
17    return 0;
18 }
19
20 int arearect (int l, int b, int n) {
21     if (n==1)
22         return l*b;
23     return arearect(l+1,b+1,n-1);
24 }
```

The terminal output shows the execution results for `n=3`, `l=5`, and `b=4`:

```
PS C:\Users\Vidhi Shah\Desktop\VITC Sem 2\C++ Lab\PPS10> gcc -o concrect concrect.c
PS C:\Users\Vidhi Shah\Desktop\VITC Sem 2\C++ Lab\PPS10> ./concrect

Number of Rectangles: 3
Length of Innermost Rectangle: 5
Breadth of Outermost Rectangle: 4
Area of Outermost Rectangle: 42 sq. units
PS C:\Users\Vidhi Shah\Desktop\VITC Sem 2\C++ Lab\PPS10>
```

```
#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 Rectangle: %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

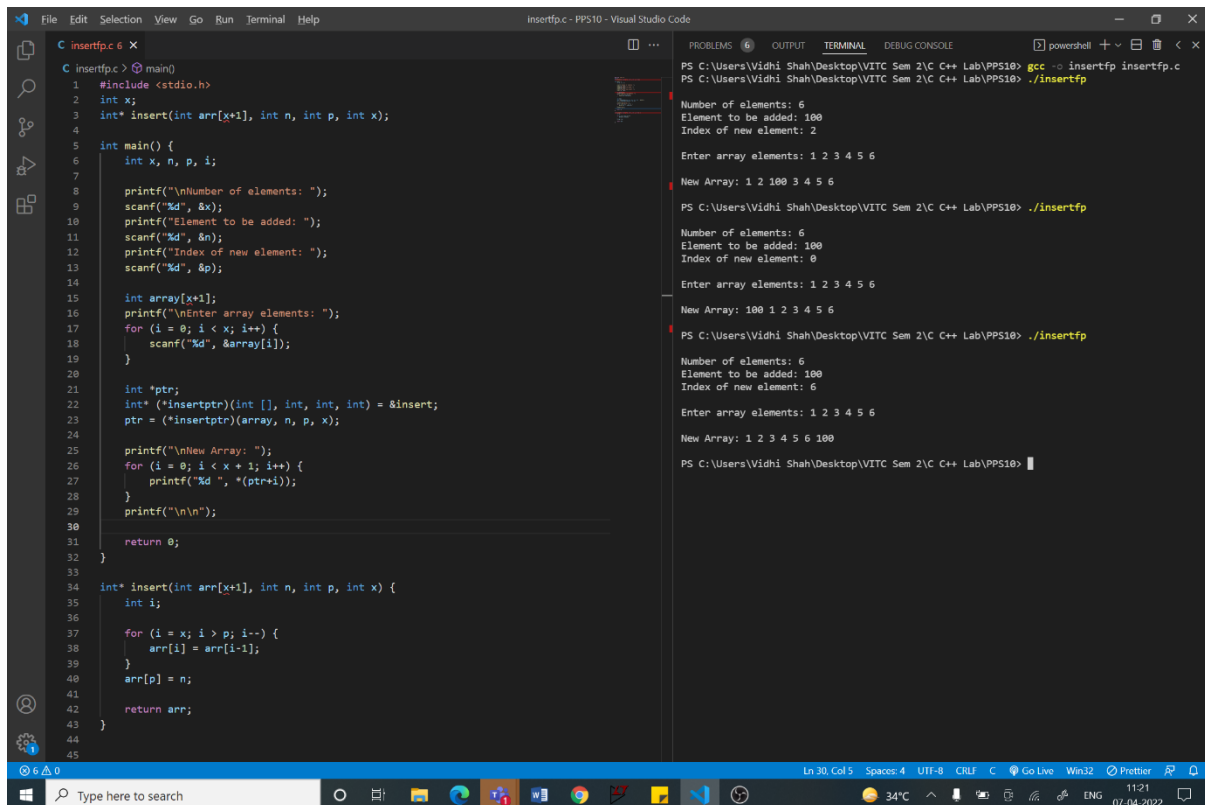
Step A: arr[i] = arr[i-1]

Step B: i = i - 1

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

Step 3: Return arr (Pointer to the array)

Code:



```
insertfp.c: X
C insertfp.c > main()
1 #include <stdio.h>
2 int x;
3 int* insert(int arr[x+1], int n, int p, int x);
4
5 int main() {
6     int x, n, p, i;
7
8     printf("\nNumber of elements: ");
9     scanf("%d", &x);
10    printf("Element to be added: ");
11    scanf("%d", &n);
12    printf("Index of new element: ");
13    scanf("%d", &p);
14
15    int array[x+1];
16    printf("\nEnter array elements: ");
17    for (i = 0; i < x; i++) {
18        scanf("%d", &array[i]);
19    }
20
21    int *ptr;
22    int* (*insertptr)(int [], int, int, int) = &insert;
23    ptr = (*insertptr)(array, n, p, x);
24
25    printf("\nNew Array: ");
26    for (i = 0; i < x + 1; i++) {
27        printf("%d ", *(ptr+i));
28    }
29    printf("\n\n");
30
31    return 0;
32 }
33
34 int* insert(int arr[x+1], int n, int p, int x) {
35     int i;
36
37     for (i = x; i > p; i--) {
38         arr[i] = arr[i-1];
39     }
40     arr[p] = n;
41
42     return arr;
43 }
44
45
```

PROBLEMS OUTPUT TERMINAL DEBUG CONSOLE

PS C:\Users\Vidhi Shah\Desktop\WITC Sem 2\C++ Lab\PPS10> gcc -o insertfp insertfp.c

PS C:\Users\Vidhi Shah\Desktop\WITC Sem 2\C++ Lab\PPS10> ./insertfp

Number of elements: 6
Element to be added: 100
Index of new element: 2
Enter array elements: 1 2 3 4 5 6
New Array: 1 2 100 3 4 5 6
PS C:\Users\Vidhi Shah\Desktop\WITC Sem 2\C++ Lab\PPS10> ./insertfp

Number of elements: 6
Element to be added: 100
Index of new element: 0
Enter array elements: 1 2 3 4 5 6
New Array: 100 1 2 3 4 5 6
PS C:\Users\Vidhi Shah\Desktop\WITC Sem 2\C++ Lab\PPS10> ./insertfp

Number of elements: 6
Element to be added: 100
Index of new element: 6
Enter array elements: 1 2 3 4 5 6
New Array: 1 2 3 4 5 6 100
PS C:\Users\Vidhi Shah\Desktop\WITC Sem 2\C++ Lab\PPS10> █

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'

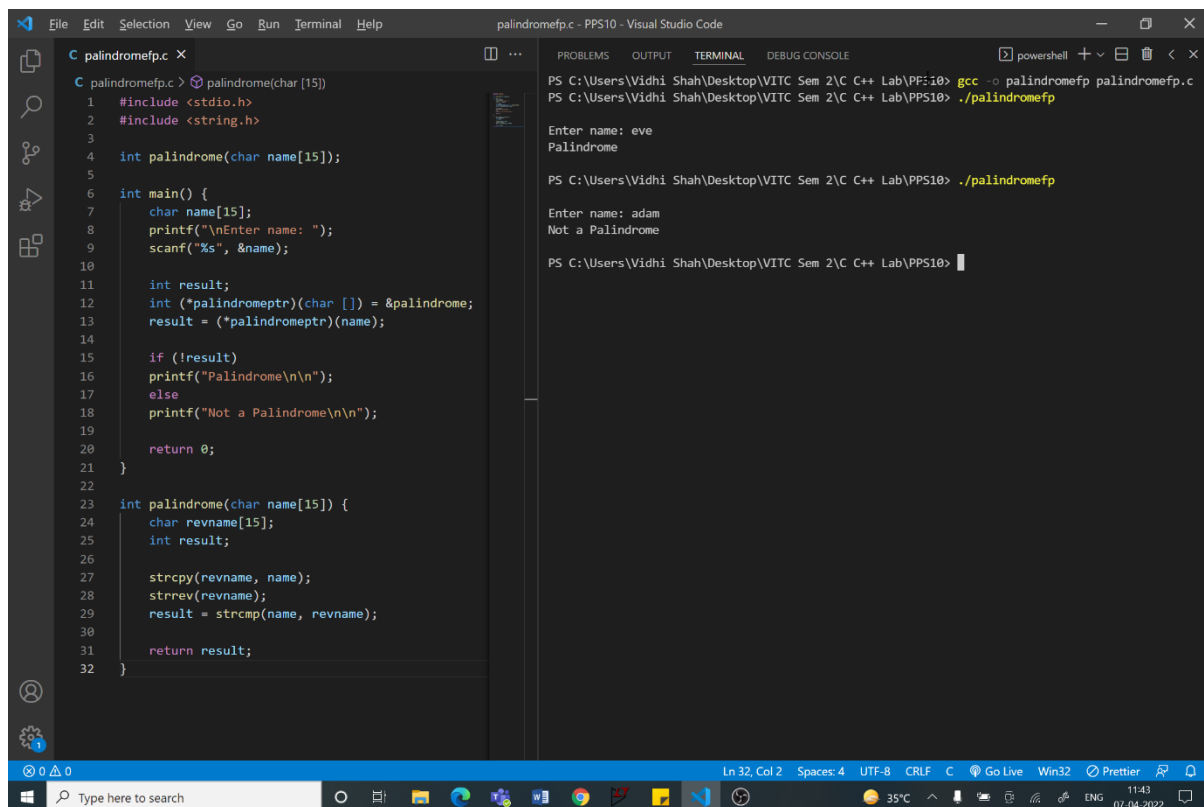
Step 2: Copy string 'name' to 'revname'

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:



The screenshot shows the Visual Studio Code editor with a C program named `palindrome.c` open. The code defines a `palindrome` function that checks if a string is a palindrome by reversing it and comparing. The `main` function prompts the user to enter a name and calls the `palindrome` function. The terminal window shows the compilation and execution of the program. The user enters 'eve' and the program outputs 'Palindrome'. The user enters 'adam' and the program outputs 'Not a Palindrome'.

```
C palindrome.c X
C palindrome.c > palindrome(char [15])
1 #include <stdio.h>
2 #include <string.h>
3
4 int palindrome(char name[15]);
5
6 int main() {
7     char name[15];
8     printf("\nEnter name: ");
9     scanf("%s", &name);
10
11     int result;
12     int (*palindromeptr)(char []) = &palindrome;
13     result = (*palindromeptr)(name);
14
15     if (!result)
16         printf("Palindrome\n\n");
17     else
18         printf("Not a Palindrome\n\n");
19
20     return 0;
21 }
22
23 int palindrome(char name[15]) {
24     char revname[15];
25     int result;
26
27     strcpy(revname, name);
28     strrev(revname);
29     result = strcmp(name, revname);
30
31     return result;
32 }
```

PS C:\Users\Vidhi Shah\Desktop\WITC Sem 2\C++ Lab\PPS10> gcc -o palindromefp palindromefp.c
PS C:\Users\Vidhi Shah\Desktop\WITC Sem 2\C++ Lab\PPS10> ./palindromefp
Enter name: eve
Palindrome
PS C:\Users\Vidhi Shah\Desktop\WITC Sem 2\C++ Lab\PPS10> ./palindromefp
Enter name: adam
Not a Palindrome
PS C:\Users\Vidhi Shah\Desktop\WITC Sem 2\C++ Lab\PPS10>

```
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;
}
```