Module Title

Principles of Programming

Weekly Assignment - Practical 5

Year

2025

Student Name: NIRVIK K.C.

UWE ID: 25024649

Assignment Submission Date: January 11, 2026

This assignment consists of the programming questions from practical exercises related to the topics from week 5.

Questions

1. (a) Create a new C program in the practical5 folder, called CalculatingMortgageA.c. The program takes as input from the keyboard two salaries and calculate the mortgage based on the two salaries.

This program solves the problem in a sequential way using the main function only, i.e. take as input two salaries, select the larger and smaller salaries, calculate the mortgage and display the mortgage. However, the program outline has already divided the program into sections, among which the selections for selecting the larger salary and the smaller salary can be defined in separate functions. These separate functions can then be called in the mortgage calculation section of the main function to replace largerSalary and smallerSalary. This is also because selecting a larger or smaller number between two given numbers is a pretty standard task that needs to be done often so if the task is done by a function, the function can be reused. Run and test your program with different pairs of salaries and see the result.

Given code:

```
/* CalculateMortgageA.c

Practical 5, Part 1 (a)

@author your name */
```

```
#include <stdio.h>
int main()
{
  //Declare variables
  double salary1, salary2, largerSalary, smallerSalary, mortgage;
  //Read in two salaries, salary1, salary2
  printf("Enter two salaries separated by a space: \n");
  scanf(" %lf %lf", &salary1, &salary2);
  //Select the larger salary, largerSalary
  if (salary1 > salary2)
    largerSalary = salary1;
  else
    largerSalary = salary2;
 //Select the smaller salary, smallerSalary
 if (salary1 > salary2)
   smallerSalary = salary2;
else
   smallerSalary = salary1;
 //Calculate mortgage

 mortgage = largerSalary * 3;

 mortgage += smallerSalary;

 Display the mortgage

 printf("The maximum size of mortgage is: £ %.2lf \n", mortgage);

}
```

Answer:

Following code for input:

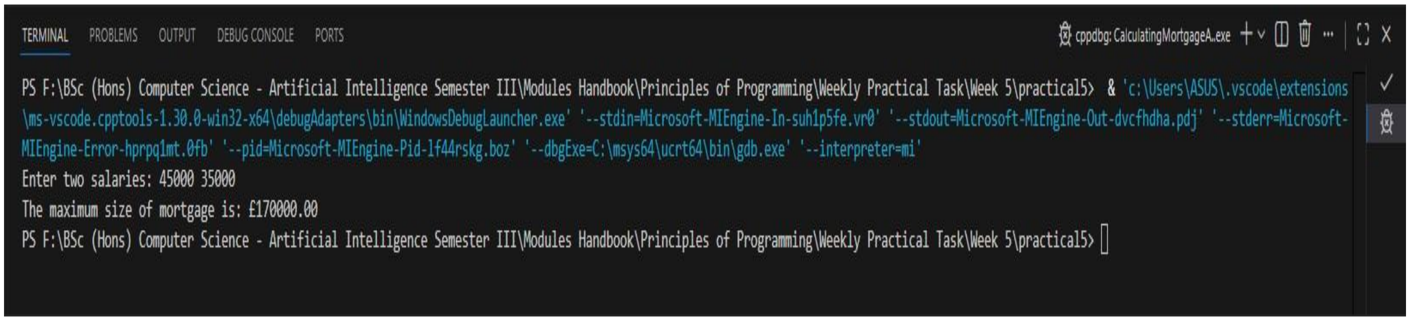
```
/* read values from input file

Practical 5 -  Part 1 (a), Functions

@Nirvik K.C. */


#include <stdio.h>

// Function prototypes

double getLarger(double a, double b);

double getSmaller(double a, double b);
```

```
int main()

{

   double salary1, salary2;

   double largerSalary, smallerSalary;

   double mortgage;

   // Read in two salaries, salary1, salary2

   printf("Enter two salaries: ");

   scanf("%lf %lf", &salary1, &salary2);


   // Get larger and smaller salaries

   largerSalary = getLarger(salary1, salary2);

   smallerSalary = getSmaller(salary1, salary2);


   // Calculate mortgage

   mortgage = largerSalary * 3;

   mortgage = mortgage + smallerSalary;

  // Display the mortgage

   printf("The maximum size of mortgage is: £%.2f\n", mortgage);
```

```
    return 0;
}
// Return the larger salary, largerSalary
double getLarger(double a, double b)
{
    if (a > b)
    {
        return a;
    }
    else
    {
        return b;
    }
}
double getSmaller(double a, double b)
{
    if (a < b)
    {
        return a;
    }
    else
    {
        return b;
    }
}


/*Output:
Enter two salaries: 45000 35000
The maximum size of mortgage is: £170000.00


Enter two salaries: 35000 50000
The maximum size of mortgage is: £185000.00
*/
```

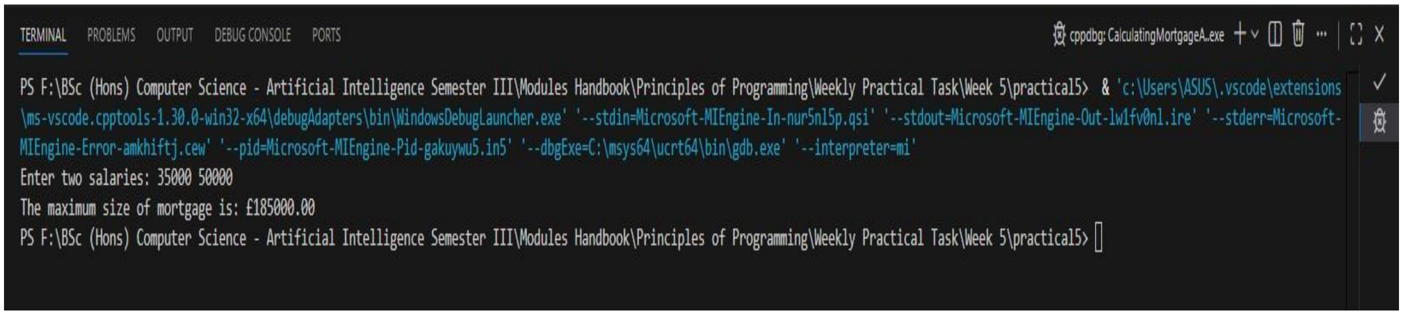
Output obtained in execution:

Example 1:



```
PS F:\BSc (Hons) Computer Science - Artificial Intelligence Semester III\Modules Handbook\Principles of Programming\Weekly Practical Task\Week 5\practical5>  & 'c:\Users\ASUS\.vscode\extensions
\ms-vscode.cpptools-1.30.0-win32-x64\debugAdapters\bin\WindowsDebugLauncher.exe' '--stdin=Microsoft-MIEngine-In-suh1p5fe.vr0' '--stdout=Microsoft-MIEngine-Out-dvcfhdha.pdj' '--stderr=Microsoft-
MIEngine-Error-hprpq1mt.0fb' '--pid=Microsoft-MIEngine-Pid-1f44rskg.boz' '--dbgExe=C:\msys64\ucrt64\bin\gdb.exe' '--interpreter=mi'
Enter two salaries: 45000 35000
The maximum size of mortgage is: £170000.00
PS F:\BSc (Hons) Computer Science - Artificial Intelligence Semester III\Modules Handbook\Principles of Programming\Weekly Practical Task\Week 5\practical5>
```

Example 2:



```
PS F:\BSc (Hons) Computer Science - Artificial Intelligence Semester III\Modules Handbook\Principles of Programming\Weekly Practical Task\Week 5\practical5>  & 'c:\Users\ASUS\.vscode\extensions
\ms-vscode.cpptools-1.30.0-win32-x64\debugAdapters\bin\WindowsDebugLauncher.exe' '--stdin=Microsoft-MIEngine-In-nur5nl5p.qsi' '--stdout=Microsoft-MIEngine-Out-lw1fv0nl.ire' '--stderr=Microsoft-
MIEngine-Error-amkhiftj.cew' '--pid=Microsoft-MIEngine-Pid-gakuywu5.in5' '--dbgExe=C:\msys64\ucrt64\bin\gdb.exe' '--interpreter=mi'
Enter two salaries: 35000 50000
The maximum size of mortgage is: £185000.00
PS F:\BSc (Hons) Computer Science - Artificial Intelligence Semester III\Modules Handbook\Principles of Programming\Weekly Practical Task\Week 5\practical5>
```

1. (b) Step 1: Create a new C program in the practical5 folder, called CalculateMortgageB.c.

Given code:

/* CalculateMortgageB.c

Practical 5, Part 2 (b)

@author your name */

Step 2. Design a program outline using functions Add the following program outline into CalculateMortgageB.c:

```
#include <stdio.h>

int main()

{

 //Declare variables

 double salary1, salary2, mortgage;

 //Read in two salaries, salary1, salary2

printf("Enter two salaries separated by a space: \n");

scanf(" %lf %lf", &salary1, &salary2);

//Calculate mortgage

      //call largerSalary function

      mortgage = largerSalary(salary1,salary2) * 3;

      //call smallerSalary function

      mortgage += smallerSalary(salary1, salary2);

//Display the mortgage

printf("The maximum size of mortgage is: £ %.2lf \n", mortgage);

}
```

As you can see there are two syntax errors indicating that the function largerSalary(double, double) is undefined and the function smallerSalary(double, double) is undefined. This is because though we have designed how to use the two functions in the program, we haven’t actually defined them. We need to define them next.

Step 3: Add function declaration and stubs into the program outline Add the following two function declarations before the main function:

```
double largerSalary(double salary1, double salary2);

double smallerSalary(double salary1, double salary2);
```

Add the following two function stubs after the main function:

```
//function stub for largerSalary(double, double)

double largerSalary(double salary1, double salary2) {

return 1;

} //function stub

//function stub for smallerSalary(double, double)

double smallerSalary(double salary1, double salary2) {

return 1;

} //function stub
```

Step 4: Run and test your program Run and test your program with different pairs of salaries and see the result.

Answer:

Reason why the result is wrong ?

When you the program, the mortgage will likely always display as £ 4.00, regardless of what salaries you enter (1 * 3 + 1 = 4).

The program is logically incomplete. The main function is correctly asking for the salaries from the user, but the functions largerSalary and smallerSalary are currently stubs. A stub is used to test the flow of a program to ensure the parts of the code are working. The functions are currently ignoring the inputs (salary1, salary2) and always "return 1", no matter what salaries are entered.

No matter what salaries you enter, the program will always give output:

The maximum size of mortgage is: £4.00

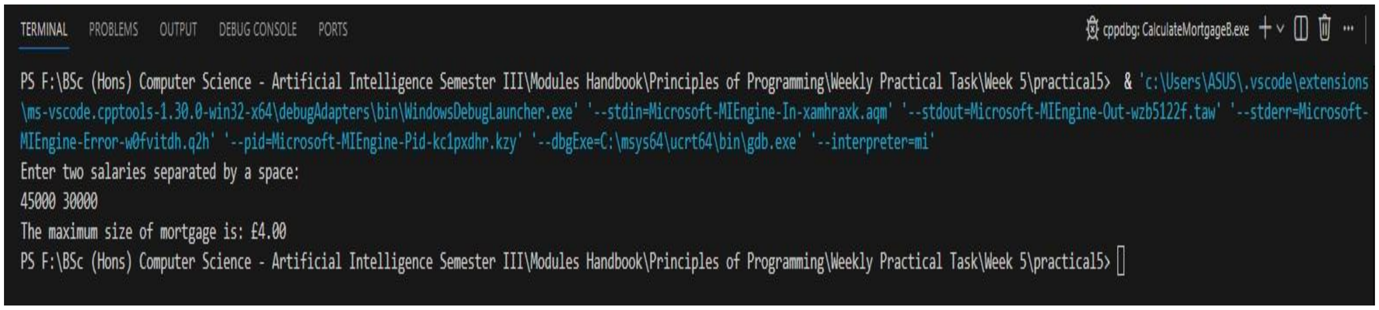
Following code for input:

```
/* read values from input file

Practical 5 -  Part 1 (b)

@Nirvik K.C. */
```

```
#include <stdio.h>

// Function prototypes declarations

double largerSalary(double salary1, double salary2);

double smallerSalary(double salary1, double salary2);


int main()

{

   // Read in two salaries, salary1, salary2

   double salary1, salary2, mortgage;

   printf("Enter two salaries separated by a space: \n");

   scanf(" %lf %lf", &salary1, &salary2);

   // Calculate mortgage

   mortgage = largerSalary(salary1, salary2) * 3;

   mortgage = mortgage + smallerSalary(salary1, salary2);

   printf("The maximum size of mortgage is: £%.2lf\n", mortgage);

   return 0;

}
```

```
// function stub for largerSalary(double, double)

double largerSalary(double salary1, double salary2)

{

   return 1; // function stub

}

// function stub for smallerSalary(double, double)

double smallerSalary(double salary1, double salary2)

{

   return 1; // function stub

}
```

Output obtained in execution:



```
PS F:\BSc (Hons) Computer Science - Artificial Intelligence Semester III\Modules Handbook\Principles of Programming\Weekly Practical Task\Week 5\practical5>  & 'c:\Users\ASUS\.vscode\extensions
\ms-vscode.cpptools-1.30.0-win32-x64\debugAdapters\bin\WindowsDebugLauncher.exe' '--stdin=Microsoft-MIEngine-In-xamhraxk.aqm' '--stdout=Microsoft-MIEngine-Out-wzb5122f.taw' '--stderr=Microsoft-
MIEngine-Error-w0fvitdh.q2h' '--pid=Microsoft-MIEngine-Pid-kc1pxdhr.kzy' '--dbgExe=C:\msys64\ucrt64\bin\gdb.exe' '--interpreter=mi'
Enter two salaries separated by a space:
45000 30000
The maximum size of mortgage is: £4.00
PS F:\BSc (Hons) Computer Science - Artificial Intelligence Semester III\Modules Handbook\Principles of Programming\Weekly Practical Task\Week 5\practical5> 
```

1. (c) Part 1 (b) does not display correct mortgage because the functions do not return the correct results. We can now actually define the two functions by implementing them.

Step 1: Create a new C program Save CalculateMortgageB.c in the practical5 folder as CalculateMortgageC.c. Change the program header where appropriate.

Step 2. Implement the function stubs to complete your program Replace the function stubs with the following function definitions:

```
//Function for selecting larger salary

double largerSalary(double salary1, double salary2) {

      if (salary1 > salary2)

          return salary1; else return salary2;

} // largeSalary

//Function for selecting smaller salary

double smallerSalary(double salary1, double salary2)

{

       if (salary1 > salary2)

          return salary2;

      else return salary1;

} // smallersalary
```

Step 3. Run and test your program Run and test your complete program with different pairs of salaries and see the result. Your program should now run and display correct mortgages.

Analysis: Congratulations. You have successfully gone through the whole process of designing, implementing, running and testing a complete program which contains multiple function definitions and calls to those methods in the main function. Now compare the three different programs we have implemented in this part and identify the main differences between them. What we have seen in this tutorial is to call functions by passing individual values to them. Once the values have been passed to a function, whatever happens within the function does not change the variables used in the main function for passing those values. In Part 2, we will define functions that take array names (i.e. pointer variables) from the function calls so we can pass array references instead of individual values to the functions when are called in the main function.

Answer:

Following code for input:

/* read values from input file

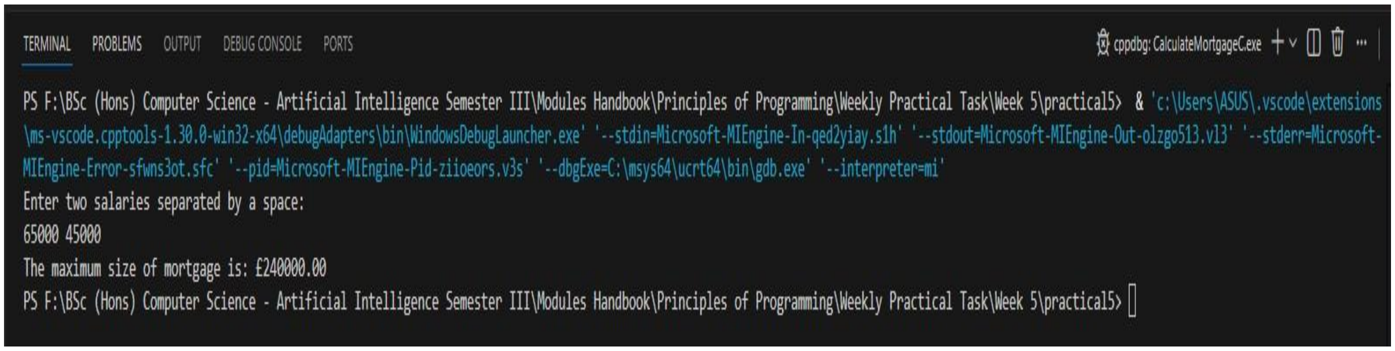
Practical 5 - Part 1 (c), changes in program Part 1 (b)

@Nirvik K.C. */

```
#include <stdio.h>
// Function prototypes declaration
double largerSalary(double salary1, double salary2);
double smallerSalary(double salary1, double salary2);


int main()
{
    double salary1, salary2, mortgage;
    // Read in two salaries from the user, salary1, salary2
    printf("Enter two salaries separated by a space: \n");
    scanf(" %lf %lf", &salary1, &salary2);
    // Calculate mortgage
    mortgage = largerSalary(salary1, salary2) * 3;
    mortgage = mortgage + smallerSalary(salary1, salary2);


    printf("The maximum size of mortgage is: £%.2lf\n", mortgage);
    return 0;
}
```

```
// Replace the function stubs
// Function for selecting larger salary
double largerSalary(double salary1, double salary2)
{
   if (salary1 > salary2)
   {
      return salary1;
   }
   else
   {
      return salary2;
   }
} // largeSalary
// Function for selecting smaller salary
double smallerSalary(double salary1, double salary2)
{
   if (salary1 > salary2)
   {
        return salary2;
    }
    else
    {
        return salary1;
    }
} // smallerSalary
/*Output:
Enter two salaries separated by a space:
65000 45000
The maximum size of mortgage is: £240000.00
*/
```

Output obtained in execution:



```
PS F:\BSc (Hons) Computer Science - Artificial Intelligence Semester III\Modules Handbook\Principles of Programming\Weekly Practical Task\Week 5\practical5>  & 'c:\Users\ASUS\.vscode\extensions\ms-vscode.cpptools-1.30.0-win32-x64\debugAdapters\bin\WindowsDebugLauncher.exe' '--stdin=Microsoft-MIEngine-In-qed2yiay.s1h' '--stdout=Microsoft-MIEngine-Out-olzgo513.vl3' '--stderr=Microsoft-MIEngine-Error-sfwns3ot.sfc' '--pid=Microsoft-MIEngine-Pid-ziioeors.v3s' '--dbgExe=C:\msys64\ucrt64\bin\gdb.exe' '--interpreter=mi'
Enter two salaries separated by a space:
65000 45000
The maximum size of mortgage is: £240000.00
PS F:\BSc (Hons) Computer Science - Artificial Intelligence Semester III\Modules Handbook\Principles of Programming\Weekly Practical Task\Week 5\practical5>
```

2. Using Functions and Pointers

The problem: Identifying a repeated offender

In a criminal investigation, DNA profiling was used to match the DNA of a suspect who has been arrested for suspicion of committing a burglary against the DNA profiling of several criminals who have been convicted of committing similar crimes. To identify a repeated offender correctly 10 particular gene chromosomes of the suspect's DNA profile were chosen which must exactly match for the corresponding ones of a convicted criminal.

The suspect and each of the convicted criminals is represented by a reference number in the range 1000 - 9999 followed by a sequence of ten real numbers (double) representing the percentage amount of ten types of gene chromosome present in the DNA profile. An input file typically contains the data for the DNA of the suspect followed on the next line by an integer specifying the number of subsequent lines in the file. Assume the data has been validated. Each of these lines contains the data for DNA of a convicted criminal. The data for each convicted criminal begins on a new line.

You are required to write a program, which will take input from the input file, match the DNA profile of the suspect with each of the convicted criminals, and display a report listing the reference number of the DNA profile which matches the DNA profile of the suspect who therefore can be identified as a repeated offender. Your program should contain a function with the following function declaration:
bool matchingProfiles(float suspect[], float aCriminal[]);

which will return true if two profiles, which are represented by suspect[] and aCriminal[] respectively, match; Otherwise return false.

Step 2. Create a text file, called dna_input.txt, and add it to the practical5 folder as the input file to your program. The text file should contain the following data:

```
2.3 3.3 4.5 6.7 7.8 2.1 3.2 4.3 5.2 6.5


5


2.3 3.3 4.5 6.7 7.8 2.1 3.2 4.3 5.2 6.5
1.3 0.3 9.5 8.7 5.8 4.1 3.2 2.3 6.2 6.9
6.3 9.3 4.3 6.4 7.5 2.9 3.0 4.1 5.3 6.5
6.1 9.4 4.5 6.6 7.4 2.8 3.2 4.4 5.0 6.0
2.3 3.3 4.5 6.6 7.8 2.2 3.2 4.3 5.2 6.5
```

Step 3: Using a text file as input file

Save RepeatedOffenderA.c in the practical4 folder as RepeatedOffenderB.c in the practical5 folder.

Step 4: Change the program outline

In the current program, we have one function, the main function, only. We now want to define a function to replace the section in the program for matching the profiles of the suspect and a criminal so the function can be called when two profiles are matched.

The current program looks like the following:

```
 /* Repeated offender

Practical 5, Part 2

@author your name */

#include <stdio.h>

#include <stdbool.h>

FILE *fp; // define a file pointer for the file

int main() {

// open the file and assign its address to file pointer

fp = fopen("dna_input.txt", "r");

int size = 10;

float suspect[size]; // declaring suspect array

int sizeR;

int sizeC = 10;
```

```
// read 10 input values into suspect array
printf("Reading the 10 gene chromosomes of the suspect \n");
for (int i = 0; i < size; i++)
  fscanf(fp, " %f", &suspect[i]);
// read the number of criminals
printf("Reading the number of criminals \n");
fscanf(fp, " %d", &sizeR);
float criminals[sizeR][sizeC]; // declaring criminals array
// read multiple profiles of 10 input values into criminals array
for (int i = 0; i < sizeR; i++) {
    printf("Read the 10 gene chromosomes of %dth criminal \n", i+1);
// read 10 input values of a criminal into criminals array
for (int j = 0; j < sizeC; j++)
    fscanf(fp, " %f", &criminals[i][j]);
}
// match the profile of the suspect with each of the criminals
for (int i = 0; i < sizeR; i++) {
bool match = true; // by default, the i+1th criminal matches
for (int j = 0; j < sizeC; j++) {
    printf(" %.1f ", suspect[j]);
}
printf(" \n");
for (int j = 0; j < sizeC; j++) {
    printf(" %.1f ", criminals[i][j])
}
printf(" \n");
for (int j = 0; j < sizeC; j++) {
    if (suspect[j] != criminals[i][j])
        match = false;
}
// display matching result
if (match)
   printf("The %dth criminal matches! \n", i+1);
else
   printf("The %dth criminal doesn't match! \n", i+1);
}
```

```
  fclose (fp); // always close files you've opened

  return 0;

}
```

In the current program, the highlighted section is used to decide whether two profiles match and assign the matching status to a Boolean variable, match. The match variable is then used to decide what matching result should be displayed in the next section of the program.

We now want to define a function to match two profiles so the new function can be called to decide what matching result is displayed. The new program with the function outline is as follows:

```
* Repeated offender

Practical 5, Part 2

@author your name */

#include <stdio.h>

#include <stdbool.h>

bool matchingProfiles(float suspect[], float aCriminal[]);

FILE *fp; // define a file pointer for the file

int main()
```

```
{
   // open the file and assign its address to file pointer
  fp = fopen("./dna_input.txt", "r");
  int size = 10;
  float suspect[size]; // declaring suspect array
  int sizeR;
  int sizeC = 10;
  // read 10 input values into suspect array
printf("Reading the 10 gene chromosomes of the suspect \n");
for (int i = 0; i < size; i++)
fscanf(fp, " %f", &suspect[i]);
// read the number of criminals
printf("Reading the number of criminals \n");
fscanf(fp, " %d", &sizeR);
float criminals[sizeR][sizeC]; // declaring criminals array


// read multiple profiles of 10 input values into criminals array
```

```
for (int i = 0; i < sizeR; i++) {
     printf("Read the 10 gene chromosomes of %dth criminal \n", i);
    // read 10 input values of a criminal into criminals array
    for (int j = 0; j < sizeC; j++) {
        fscanf(fp, " %f", &criminals[i][j]);
    }
// match suspect with each criminal and display the result
  for (int i = 0; i < sizeR; i++) {
       // call function, matchingProfiles() with suspect array and ith row
      // of crinimals array
     // if (matchingProfiles(suspect, criminals[i])) // function call
           printf("The %dth criminal matches! \n", i+1);
     else
          printf("The %dth criminal doesn't match! \n", i+1);
  fclose (fp); // always close files you've opened
  return 0;
}
```

```
/*****************************************************************
/ // define the function matchingProfiles(suspect[], aCriminal[])
bool matchingProfiles(float suspect[], float aCriminal[]) {
/* code */
}
```

Explanation: As you can see in the program outline above, as highlighted, once the function is defined below the main function, it can be called in the main function. Note that we also need to add a function declaration before the main function, as highlighted.

Analysis: As you can see the matchingProfiles(float suspect[], float criminal[])function takes as input two one-dimensional arrays from the main function when it is called, one for the profile of the suspect and another for the profile of the ith criminal (i.e. the ith row in the two-dimensional array, criminals[][]). Remember that each row of a two-dimensional array is itself a one-dimensional array, in this case, identified by criminals[i].

Step 5: Implement the program outline in Step 4.

Add the code to the function definition. Tip: Refer to the highlighted in repeatedOffenderA.c for matching the profile of the suspect with each of the criminals.

Step 6: Run and test the program

Once you have run and tested your program successfully. Show your working program to your tutor.

Step 7. Identify pointer variables/arguments/parameters and non-pointer variables in the above program. Explain the reasons. What are the main differences between pointer variables and non-pointer variables.

Analysis: In repeatedOffenderB.c, we have seen an example of pass by address (i.e. pass an address of an array to a function). In the function call, matchingProfiles(suspect, criminals[i]), in the main function, criminals[i] is the address of the ith row in the criminals[][] array, which is passed as an argument to the corresponding parameters in the function. As a result, whatever change made to the ith row within the function will also be made to the array in the main function. If you are interested in finding out, you can add a few lines of code to see how pass by address actually works.

Answer:

Following code for input:

```
/* read values from input file

Practical 5 -  Part 2, Using Functions and Pointers

@Nirvik K.C. */

#include <stdio.h>

#include <stdbool.h>

// Function prototype declarations
```

```
bool matchingProfiles(float suspect[], float aCriminal[]);
FILE *fp; //  defined a file pointer for the file
int main()
{
   // Open the input file
   fp = fopen("dna_input.txt", "r");
   if (fp == NULL)
   {
      printf("Error: Could not open file dna_input.txt\n");
      return 1;
   }
   int size = 10;
   float suspect[size]; // declaring suspect array
   int sizeR;
   int sizeC = 10;
   // Read 10 input values into suspect array
   printf("Reading the 10 gene chromosomes of the suspect \n");
```

```
 for (int i = 0; i < size; i++)
{
   fscanf(fp, " %f", &suspect[i]);
}


// Read the number of criminals
printf("Reading the number of criminals \n");
fscanf(fp, " %d", &sizeR);
float criminals[sizeR][sizeC]; // declaring criminals array
// read multiple profiles of 10 input values into criminals array
for (int i = 0; i < sizeR; i++)
{
   printf("Read the 10 gene chromosomes of %dth criminal \n", i + 1);
   for (int j = 0; j < sizeC; j++)
   {
      fscanf(fp, " %f", &criminals[i][j]);
   }
}
```

```
// match suspect with each criminal and display the result

printf("DNA matching report of the suspect with criminals:\n");

bool foundAnyMatch = false;

for (int i = 0; i < sizeR; i++)

{

   if (matchingProfiles(suspect, criminals[i]))

   {

      printf("The %dth criminal matches! \n", i + 1);

      foundAnyMatch = true;

   }

   else

   {

      printf("The %dth criminal doesn't match! \n", i + 1);

   }

}
```

```
    if (foundAnyMatch)
    {
        printf("The suspect is a repeated offender! \n");
    }
    else
    {
        printf("The suspect is not a repeated offender! \n");
    }
    fclose(fp);
    return 0;
}

bool matchingProfiles(float suspect[], float aCriminal[])
{
    for (int i = 0; i < 10; i++)
    {
        if (suspect[i] != aCriminal[i])
        {
            return false; // mismatch found
        }
    }
    return true; // all values matched
}


/*Output:
Reading the 10 gene chromosomes of the suspect
Reading the number of criminals
Read the 10 gene chromosomes of 1th criminal
Read the 10 gene chromosomes of 2th criminal
Read the 10 gene chromosomes of 3th criminal
Read the 10 gene chromosomes of 4th criminal
Read the 10 gene chromosomes of 5th criminal
DNA matching report of the suspect with criminals:
The 1th criminal matches!
The 2th criminal doesn't match!
The 3th criminal doesn't match!
```

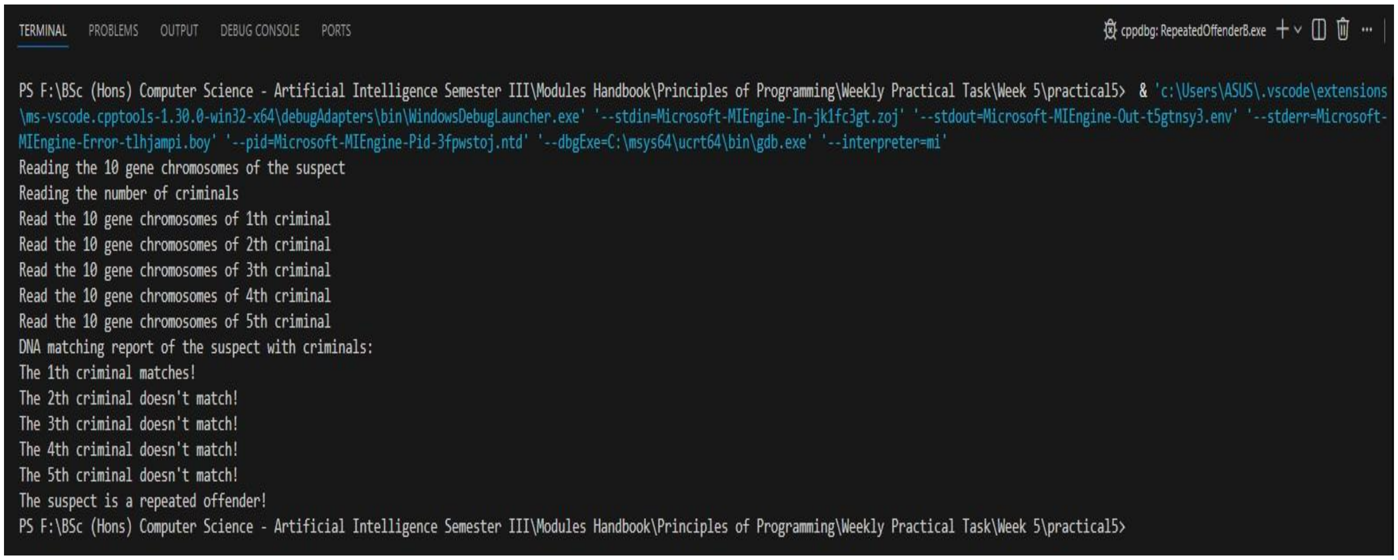
The 4th criminal doesn't match!

The 5th criminal doesn't match!

The suspect is a repeated offender!

*/

Output obtained in execution:



```
PS F:\BSc (Hons) Computer Science - Artificial Intelligence Semester III\Modules Handbook\Principles of Programming\Weekly Practical Task\Week 5\practical5>  & 'c:\Users\ASUS\.vscode\extensions\ms-vscode.cpptools-1.30.0-win32-x64\debugAdapters\bin\WindowsDebugLauncher.exe' '--stdin=Microsoft-MIEngine-In-jk1fc3gt.zoj' '--stdout=Microsoft-MIEngine-Out-t5gtnsy3.env' '--stderr=Microsoft-MIEngine-Error-tlhjampi.boy' '--pid=Microsoft-MIEngine-Pid-3fpwstoj.ntd' '--dbgExe=C:\msys64\ucrt64\bin\gdb.exe' '--interpreter=mi'
Reading the 10 gene chromosomes of the suspect
Reading the number of criminals
Read the 10 gene chromosomes of 1th criminal
Read the 10 gene chromosomes of 2th criminal
Read the 10 gene chromosomes of 3th criminal
Read the 10 gene chromosomes of 4th criminal
Read the 10 gene chromosomes of 5th criminal
DNA matching report of the suspect with criminals:
The 1th criminal matches!
The 2th criminal doesn't match!
The 3th criminal doesn't match!
The 4th criminal doesn't match!
The 5th criminal doesn't match!
The suspect is a repeated offender!
PS F:\BSc (Hons) Computer Science - Artificial Intelligence Semester III\Modules Handbook\Principles of Programming\Weekly Practical Task\Week 5\practical5>
```

Step 7. Identify pointer variables/arguments/parameters and non-pointer variables in the above program. Explain the reasons. What are the main differences between pointer variables and non-pointer variables.

Answer:

Pointer variables: fp variable

Reason: Explicit pointer to a FILE structure which is returned by “fopen”.

Pointer Parameters: float suspect[ ], float aCriminal[ ]

Reason: When an array is passed to a function, it acts as a reference or simplifies into a pointer to its first element. The function receives the address, not copy of the whole array.

Pointer arguments: suspect, criminals[i]

Reason: When used in the function call matchingProfiles(suspect, criminals[i]), these arguments represent the starting memory addresses of the respective data sets.

Non-pointer Variables: size, sizeR, sizeC, i, j, foundAnyMatch

Reason: These variables store actual values (numeric) directly in their assigned memory slots. The “foundAnyMatch” is a bool type variable which stores the actual data (True or False) directly in memory.

Non-pointer values: suspect[i], criminals[i][j]

Reason: The specific indexed element represents a single float value, not an address.

Differences between pointer variables and non-pointer variables:

Pointer Variables	Non-pointer Variables
1) Pass-by-address is the passing mechanism for such type of variables where only memory address is passed.	1) Pass-by-value is the passing mechanism for such type of variables where a copy of it is made.
2) Changes inside the function do affect original variable.	2) Changes inside the function do not affect original variable.
3) It is typically used for large data (arrays), need to modify original strings, and structures.	3) It is typically used for small/simple data, control values, and counters.
4) Examples: size, sizeR, I, foundAnyMatch, etc.	4) Examples: suspect[], aCriminal [], fp, criminals[i],etc.

3. More Programming exercises

3. (i) (Sort three numbers)

Write a function with the following function declaration to

display three numbers in increasing order:

int DisplaySortedNumbers(double num1, double num2, double num3);

Now write a test program that prompts the user to enter three numbers and invokes the function to display them in increasing order.

Answer:

Following code for input:

```
/* read values from input file
Practical 5 -  Part 3, 5.1 (Sort three numbers)
@Nirvik K.C. */
#include <stdio.h>
void DisplaySortedNumbers(double num1, double num2, double num3)
{
   double smallestNum, middleNum, largestNum;
   // Determine the smallest number
   if (num1 <= num2 && num1 <= num3)
   {
      smallestNum = num1;
   }
   else if (num2 <= num1 && num2 <= num3)
   {
      smallestNum = num2;
   }
```

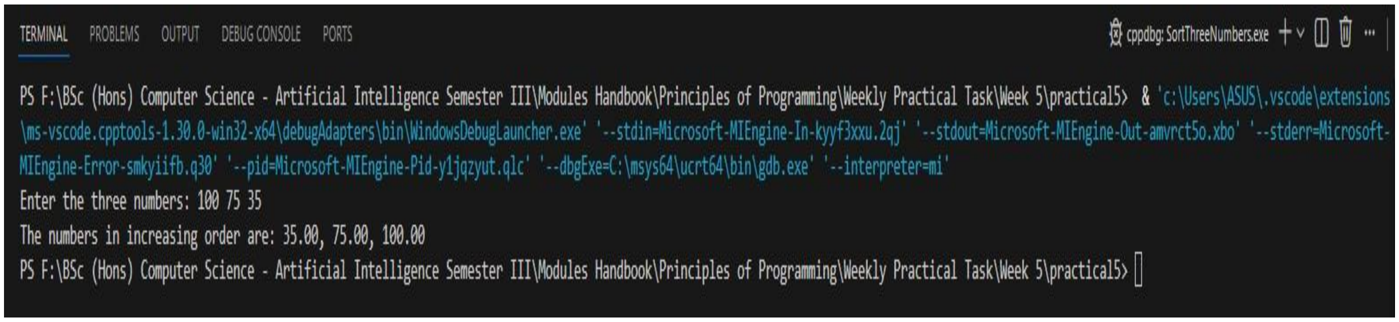
```
 else
{
   smallestNum = num3;
}
// Determine the largest number
if (num1 >= num2 && num1 >= num3)
{
   largestNum = num1;
}
else if (num2 >= num1 && num2 >= num3)
{
   largestNum = num2;
}
else
{
   largestNum = num3;
}
```

```
   // Determine the number middle number (neither smallest nor largest)
   if (num1 != smallestNum && num1 != largestNum)
   {
      middleNum = num1;
   }
   else if (num2 != smallestNum && num2 != largestNum)
   {
      middleNum = num2;
   }
   else
   {
      middleNum = num3;
   }
   // Display the three numbers in increasing order
   printf("%.2lf, %.2lf, %.2lf\n", smallestNum, middleNum, largestNum);
}
```

```
int main()
{
   double a, b, c;
   printf("Enter the three numbers: ");
   scanf(" %lf %lf %lf", &a, &b, &c);
   printf("The numbers in increasing order are: ");
   DisplaySortedNumbers(a, b, c);
   return 0;
}


/*Output:
Enter the three numbers: 100 75 35
The numbers in increasing order are: 35.00, 75.00, 100.00


Enter the three numbers: 7.5 2.3 4.0
The numbers in increasing order are: 2.30, 4.00, 7.50
*/
```

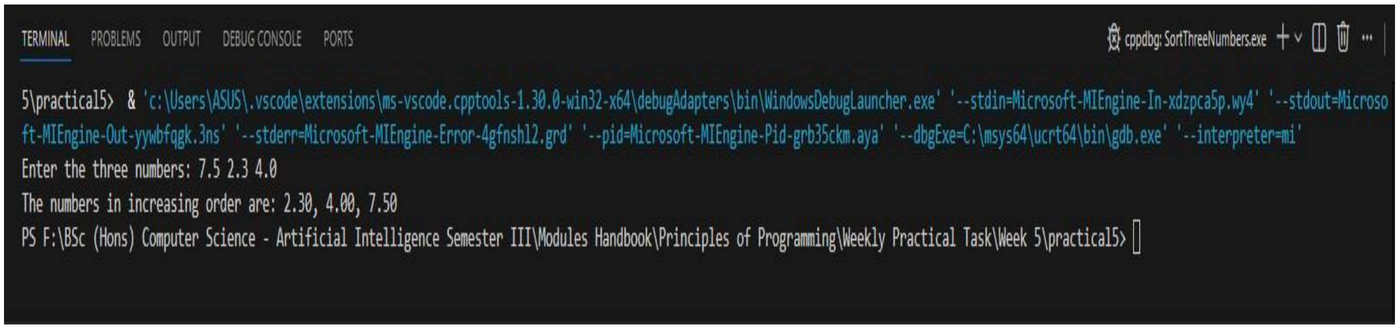
Output obtained in execution:

Example 1:



```
PS F:\BSc (Hons) Computer Science - Artificial Intelligence Semester III\Modules Handbook\Principles of Programming\Weekly Practical Task\Week 5\practical5>  & 'c:\Users\ASUS\.vscode\extensions
\ms-vscode.cpptools-1.30.0-win32-x64\debugAdapters\bin\WindowsDebugLauncher.exe' '--stdin=Microsoft-MIEngine-In-kyyf3xxu.2qj' '--stdout=Microsoft-MIEngine-Out-amvrct5o.xbo' '--stderr=Microsoft-
MIEngine-Error-smkyiifb.q30' '--pid=Microsoft-MIEngine-Pid-y1jqzyut.qlc' '--dbgExe=C:\msys64\ucrt64\bin\gdb.exe' '--interpreter=mi'
Enter the three numbers: 100 75 35
The numbers in increasing order are: 35.00, 75.00, 100.00
PS F:\BSc (Hons) Computer Science - Artificial Intelligence Semester III\Modules Handbook\Principles of Programming\Weekly Practical Task\Week 5\practical5>
```

Example 2:



```
5\practical5>  & 'c:\Users\ASUS\.vscode\extensions\ms-vscode.cpptools-1.30.0-win32-x64\debugAdapters\bin\WindowsDebugLauncher.exe' '--stdin=Microsoft-MIEngine-In-xdzpca5p.wy4' '--stdout=Microso
ft-MIEngine-Out-yywbfqgk.3ns' '--stderr=Microsoft-MIEngine-Error-4gfnshl2.grd' '--pid=Microsoft-MIEngine-Pid-grb35ckm.aya' '--dbgExe=C:\msys64\ucrt64\bin\gdb.exe' '--interpreter=mi'
Enter the three numbers: 7.5 2.3 4.0
The numbers in increasing order are: 2.30, 4.00, 7.50
PS F:\BSc (Hons) Computer Science - Artificial Intelligence Semester III\Modules Handbook\Principles of Programming\Weekly Practical Task\Week 5\practical5>
```

3. (ii) (Check password)

Some websites impose certain rules for passwords. Write a function that checks whether a string is a valid password. Suppose the password rules are as follows: • A password must have at least eight characters. • A password consists of only letters and digits. • A password must contain at least two digits. Write a program that prompts the user to enter a password and displays Valid Password if the rules are followed or Invalid Password otherwise.

Answer:

Following code for input:

```
/* read values from input file

Practical 5 -  Part 3, 5.2 (Check password)

@Nirvik K.C. */

#include <stdio.h>

#include <string.h>

#include <stdbool.h>

#include <ctype.h>
```

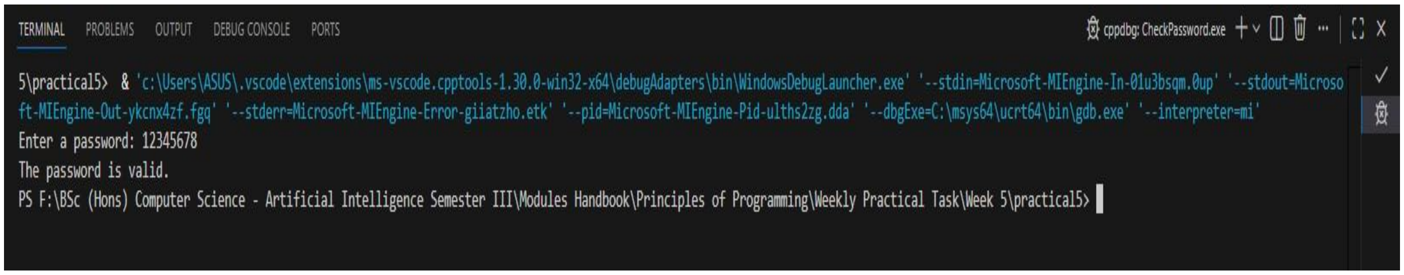
```
// Function to check password validity
int isValidPassword(const char *password)
{
   int length = strlen(password);


  // First rule: Must have at least 8 characters
   if (length < 8)
   {
      return false;
   }
    int digitCount = 0;
   // Check each character
   for (int i = 0; i < length; i++)
   {
      char ch = password[i];
      // Second Rule: Only letters and digits allowed
      if (!(
              (ch >= 'A' && ch <= 'Z') ||
              (ch >= 'a' && ch <= 'z') ||
              (ch >= '0' && ch <= '9')))
      {
          return false;
      }
      // Count the number of digits
      if (ch >= '0' && ch <= '9') {
          digitCount++;
      }
   }
   // Third Rule: Must contain at least two digits
   if (digitCount < 2)
   {
      return false;
   }
```

```
    else {
        return true; // password is valid
    }
}
int main() {
    char password[100];
    printf(“Enter a password: “);
    scanf(“%s”, password);
    if (isValidPassword(password))
    {
        printf(“The password is valid.\n”);
    }
    else
    {
        printf(“The password is invalid.\n”);
    }
    return 0;
}
```

```
/*Output:

Enter a password: 12345678

The password is valid.


Enter a password: Valid123

The password is valid.


Enter a password: Password1

The password is invalid.

*/
```

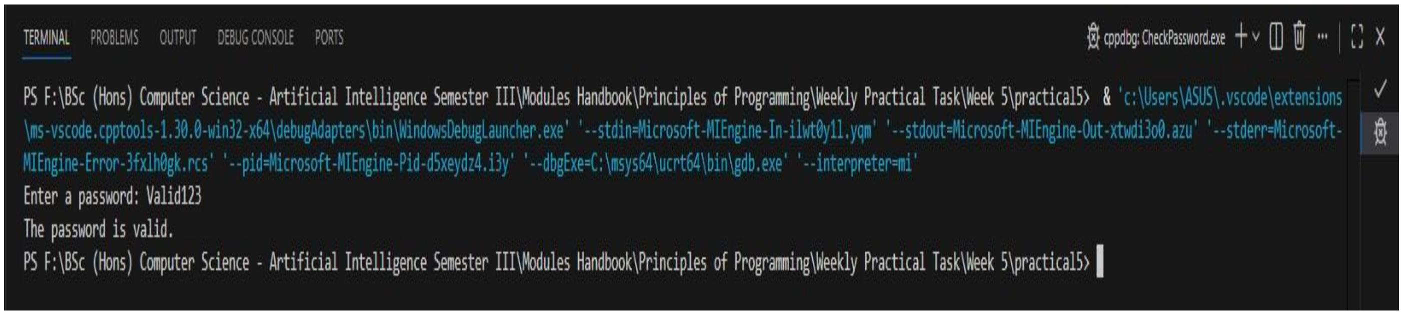
Output obtained in execution:

Example 1:



```
5\practical5> & 'c:\Users\ASUS\.vscode\extensions\ms-vscode.cpptools-1.30.0-win32-x64\debugAdapters\bin\WindowsDebugLauncher.exe' '--stdin=Microsoft-MIEngine-In-01u3bsqm.0up' '--stdout=Microso
ft-MIEngine-Out-ykcnx4zf.fgq' '--stderr=Microsoft-MIEngine-Error-giiatzho.etk' '--pid=Microsoft-MIEngine-Pid-ulths2zg.dda' '--dbgExe=C:\msys64\ucrt64\bin\gdb.exe' '--interpreter=mi'
Enter a password: 12345678
The password is valid.
PS F:\BSc (Hons) Computer Science - Artificial Intelligence Semester III\Modules Handbook\Principles of Programming\Weekly Practical Task\Week 5\practical5>
```

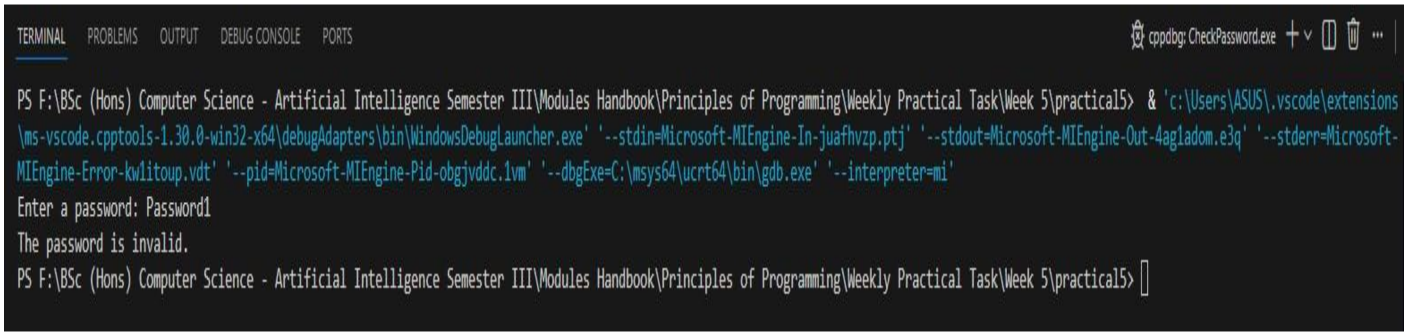
Example 2:



TERMINAL PROBLEMS OUTPUT DEBUG CONSOLE PORTS

```
PS F:\BSc (Hons) Computer Science - Artificial Intelligence Semester III\Modules Handbook\Principles of Programming\Weekly Practical Task\Week 5\practical5>  & 'c:\Users\ASUS\.vscode\extensions\ms-vscode.cpptools-1.30.0-win32-x64\debugAdapters\bin\WindowsDebugLauncher.exe' '--stdin=Microsoft-MIEngine-In-ilwt0y1l.yqm' '--stdout=Microsoft-MIEngine-Out-xtwdi3o0.azu' '--stderr=Microsoft-MIEngine-Error-3fxlh0gk.rcs' '--pid=Microsoft-MIEngine-Pid-d5xeydz4.i3y' '--dbgExe=C:\msys64\ucrt64\bin\gdb.exe' '--interpreter=mi'
Enter a password: Valid123
The password is valid.
PS F:\BSc (Hons) Computer Science - Artificial Intelligence Semester III\Modules Handbook\Principles of Programming\Weekly Practical Task\Week 5\practical5>
```

Example 3:



TERMINAL PROBLEMS OUTPUT DEBUG CONSOLE PORTS

```
PS F:\BSc (Hons) Computer Science - Artificial Intelligence Semester III\Modules Handbook\Principles of Programming\Weekly Practical Task\Week 5\practical5>  & 'c:\Users\ASUS\.vscode\extensions\ms-vscode.cpptools-1.30.0-win32-x64\debugAdapters\bin\WindowsDebugLauncher.exe' '--stdin=Microsoft-MIEngine-In-juafhvzp.ptj' '--stdout=Microsoft-MIEngine-Out-4ag1adom.e3q' '--stderr=Microsoft-MIEngine-Error-kw1itoup.vdt' '--pid=Microsoft-MIEngine-Pid-obgjvddc.1vm' '--dbgExe=C:\msys64\ucrt64\bin\gdb.exe' '--interpreter=mi'
Enter a password: Password1
The password is invalid.
PS F:\BSc (Hons) Computer Science - Artificial Intelligence Semester III\Modules Handbook\Principles of Programming\Weekly Practical Task\Week 5\practical5>
```