|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  |  | | --- | --- | | For Examiner’s Use | | | Examiner’s Initials | | | Question | Mark | | **1** |  | | **2** |  | | **3** |  | | **4** |  | | **5** |  | | **6** |  | | **7** |  | | **8** |  | | **9** |  | | **10** |  | | **11** |  | | **12** |  | | **13** |  | | **14** |  | | **15** |  | | **16** |  | | **17** |  | | **18** |  | | **TOTAL** |  |  |  |  |  |  |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | | **Centre number** |  |  |  |  |  | **Candidate number** |  |  |  |  | | **Surname** | Palmer | | | | | | | | | | | **Forename(s)** | Kenan | | | | | | | | | | | **Candidate signature** |  | | | | | | | | | | | **Programming Language** | **PYTHON (7516/1D)** | | | | | | | | | |   **AS COMPUTER SCIENCE** |
| **Paper 1** |

TBC

Time allowed: 2 hour 15 minutes

Instructions

* This is the Electronic Answer Document (EAD). Answer **all** questions by entering your answers into this document on screen. You **must save** this document at regular intervals.
* Before the examination begins, type the information needed in the boxes **at the top of this page**.
* Before the examination begins, type the information needed in the boxes **in the footers** (page 2 onwards) of this EAD.

**During the examination**

* You may print pages of your EAD. A print monitor will collect and deliver your print-out to you. You must **not** collect your own print-out.

**Exceptions**

* If you experience difficulty inserting screen shots into your EAD then you may print these separately and attach to the back of the EAD with a reference in the correct place in the EAD. Ensure that your **Centre Number**, **Candidate Name** and **Candidate Number** are on each sheet.

**At the end of the examination**

* Save for the last time and print your EAD on one side only (not double-sided). A print monitor will collect and deliver your print-out to you. Check that your details are in the footers of every page. Write them in if they are not.
* **Enter your signature on the front cover.**
* **Staple or tie all pages together in the top left-hand corner of the EAD.**
* **Hand in all pages of the EAD to the Invigilator.**

**Warning**

* No extra time is allowed for printing and collating.
* It may not be possible to credit an answer if your:

– details are not printed on every page as instructed above

– screen captures are not legible to the Examiner.

Answer **all** questions.

You **must save** this document at regular intervals or you may lose your work.

**Section A**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | | | | | | *Leave blank* |
| **Question 01** | | | | |  |  |
| **0** | **1** | **.** | **1** | {c,bc,bbc,bbbc,any number of bs followed by 1 c } | |  |
| **0** | **1** | **.** | **2** | {c,bc, 0 or 1 b followed by a. c) | |  |
| **0** | **1** | **.** | **3** | {bcbcbcbcbcbcd 1 or more b or c in any pattern followed by a d | |  |
| **0** | **1** | **.** | **4** | B(cd)\*(e|(fg)) | |  |
|  |  |  |  |  | |  |
|  |  |  |  |  | | **5** |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | | | | | | *Leave blank* |
| **Question 02** | | | | |  |  |
| **0** | **2** | **.** | **1** | |  |  |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | --- | --- | | ListLength | New | p | q | [1] | [2] | [3] | [4] | [5] | | 4 | 38 | - | - | 9 | 21 | 49 | 107 |  | | 4 | 38 | 1 | - | 9 | 21 | 49 | 107 |  | | 4 | 38 | 2 | - | 9 | 21 | 49 | 107 |  | | 4 | 38 | 3 | - | 9 | 21 | 49 | 107 |  | | 4 | 38 | 3 | 4 | 9 | 21 | 49 | 107 | 107 | | 4 | 38 | 3 | 3 | 9 | 21 | 49 | 49 | 107 | | 4 | 38 | 3 | 3 | 9 | 21 | 38 | 49 | 107 | | |  |
| **0** | **2** | **.** | **2** | Insert a number into a list in its ordered place, smallest to largest | |  |
|  |  |  |  |  | |  |
|  |  |  |  |  | | **5** |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | | | | | | *Leave blank* |
| **Question 03** | | | | |  |  |
| **0** | **3** | **.** | **1** | # Please fill cell instead of using arrow for current position   |  |  |  |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | | 1. | 1 | 0 | 1 |  |  |  | … |  | Sb |  |  |  |  |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | | 2. |  | 0 | 1 |  |  |  | … |  | S1 |  |  |  |  |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | | 3. |  | 0 | 1 |  |  |  | … |  | S1 |  |  |  |  |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | | 4. |  | 0 | 1 |  |  |  | … |  | S1 |  |  |  |  |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | | 5. |  | 0 | 1 |  |  |  | … |  | Sc1 |  |  |  |  |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | | 6. |  | 0 |  |  |  |  | … |  | Sl |  |  |  |  |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | | 7. |  | 0 |  |  |  |  | … |  | Sl |  |  |  |  |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | | 8. |  | 0 |  |  |  |  | … |  | Sb |  |  |  |  |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | | 9. |  |  |  |  |  |  | … |  | S0 |  |  |  |  |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | | 10. |  |  |  |  |  |  | … |  | Sc0 |  |  |  |  |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | | 11. |  |  |  |  |  |  | … |  | Sy | | |  |
| **0** | **3** | **.** | **2** | They go along the tape until it comes to the end of the sequence of 1s and 0s | |  |
| **0** | **3** | **.** | **3** | It could tackle any computable problem since it has infinite tape and infinite time | |  |
|  |  |  |  |  | |  |
|  |  |  |  |  | | **9** |

**Section B**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | | | | | | *Leave blank* |
| **Question 04** | | | | |  |  |
| **0** | **4** | **.** | **1** | Task 1 Code:  def convertBinary(num):  current = num  binary = []  finished = False  while not finished:  binary.append(current%2)  current = current//2  if current == 0:  finished = True  print(num)  print(binary)  num = int(input("enter a digit"))  convertBinary(num)  Task 2 Code:  def convertBinary(num):  current = num  binary = []  finished = False  while not finished:  binary.append(current%2)  current = current//2  if current == 0:  finished = True  reverse = []  for index in range(len(binary)):  reverse.append(binary[len(binary)-1-index])  print(num)  print(binary)  print(reverse)  num = int(input("enter a digit"))  convertBinary(num) | |  |
| **0** | **4** | **.** | **2** | A picture containing drawing  Description automatically generated | |  |
|  |  |  |  |  | |  |
|  |  |  |  |  | | **14** |

**Section C**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | | | | | | *Leave blank* |
| **Question 05** | | | | |  |  |
| **0** | **5** | **.** | **1** | FileFound | |  |
| **0** | **5** | **.** | **2** | ListEmpty | |  |
|  |  |  |  |  | |  |
|  |  |  |  |  | | **2** |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | | | | | | *Leave blank* |
| **Question 06** | | | | |  |  |
| **0** | 6 |  |  | MoveRecord | |  |
|  |  |  |  |  | |  |
|  |  |  |  |  | | **1** |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | | | | | | *Leave blank* |
| **Question 07** | | | | |  |  |
| **0** | 7 |  |  | To make sure that if there isn’t a file found that the program doesn’t crash and just ends | |  |
|  |  |  |  |  | |  |
|  |  |  |  |  | | **1** |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | | | | | | *Leave blank* |
| **Question 08** | | | | |  |  |
| **0** | 8 |  |  | A is passed through, so it would contain the postion and state (dame or not a dame) of each piece belonging to player A in a 2D list were each list is a piece and 3 elements in those list are the column, row and state | |  |
|  |  |  |  |  | |  |
|  |  |  |  |  | | **1** |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | | | | | | *Leave blank* |
| **Question 09** | | | | |  |  |
| **0** | **9** |  |  | |  | | --- | | 1. Player A can’t move | | (b)  Player b moves | | ( c) Player B’s turn | | (d) player B can’t move | | |  |
|  |  |  |  |  | |  |
|  |  |  |  |  | | **2** |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | | | | | | *Leave blank* |
| **Question 10** | | | | |  |  |
| **1** | **0** | **.** | **1** | DisplayBoard | |  |
| **1** | **0** | **.** | **2** | PrintResult | |  |
| **1** | **0** | **.** | **3** | PrintMiddleRow | |  |
|  |  |  |  |  | |  |
|  |  |  |  |  | | **3** |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | | | | | | *Leave blank* |
| **Question 11** | | | | |  |  |
| **1** | **1** | **.** | **1** | The 1st one is the number of moves made  The 2nd one is the number of dames the player has | |  |
| **1** | **1** | **.** | **2** | Because there are 12 pieces belonging to a | |  |
| **1** | **1** | **.** | **3** | The values stored in there relate to the column, row and whether or not the piece is a ‘dame’ | |  |
|  |  |  |  |  | |  |
|  |  |  |  |  | | **5** |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | | | | | | *Leave blank* |
| **Question 12** | | | | |  |  |
| **1** | **2** |  |  | It creates a checkered pattteren by only using the cells where the column and row numbers are odd, so ever other column is not used | |  |
|  |  |  |  |  | |  |
|  |  |  |  |  | | **3** |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | | | | | | *Leave blank* |
| **Question 13** | | | | |  |  |
| **1** | **3** |  |  | It stores the number of moves the player can make so it can be used in count loops later on | |  |
|  |  |  |  |  | |  |
|  |  |  |  |  | | **2** |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | | | | | | *Leave blank* |
| **Question 14** | | | | |  |  |
| **1** | **4** |  |  | It takes asks the user to input a piece, it will then iterate through ListOfMoves to see if it can find that move, if it is found it terminates, if not it displayes an error code | |  |
|  |  |  |  |  | |  |
|  |  |  |  |  | | **5** |

**Section D**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | | | | | | *Leave blank* |
| **Question 15** | | | | |  |  |
| **1** | **5** | **.** | **1** | def DisplayErrorCode(ErrorNumber):  if ErrorNumber == 1:  ErrorCode = " invalid piece selected."  elif ErrorNumber == 2:  ErrorCode = " that piece cant make that move."  elif ErrorNumber == 3:  ErrorCode = " invalid column or row enter."  elif ErrorNumber == 4:  ErrorCode = " file not found."  else:  ErrorCode = " unknown error "  print('Error ', ErrorNumber, ErrorCode) | |  |
| **1** | **5** | **.** | **2** | A close up of text on a black background  Description automatically generated | |  |
|  |  |  |  |  | |  |
|  |  |  |  |  | | **4** |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | | | | | | *Leave blank* |
| **Question 16** | | | | |  |  |
| **1** | **6** | **.** | **1** | def ValidJump(Board, PlayersPieces, Piece, NewRow, NewColumn):  Valid = False  MiddlePiece = ''  Player = Piece[0].lower()  Index = int(Piece[1:])  if Player == 'a':  OppositePiecePlayer = 'b'  else:  OppositePiecePlayer = 'a'  if NewRow in range(BOARD\_SIZE) and NewColumn in range(BOARD\_SIZE):  if Board[NewRow][NewColumn] == SPACE:  CurrentRow = PlayersPieces[Index][ROW]  CurrentColumn = PlayersPieces[Index][COLUMN]  MiddlePieceRow = (CurrentRow + NewRow) // 2  MiddlePieceColumn = (CurrentColumn + NewColumn) // 2  MiddlePiece = Board[MiddlePieceRow][MiddlePieceColumn]  MiddlePiecePlayer = MiddlePiece[0].lower()  if MiddlePiecePlayer == OppositePiecePlayer :  Valid = True  return Valid | |  |
| **1** | **6** | **.** | **2** | A screenshot of a cell phone  Description automatically generated | |  |
|  |  |  |  |  | |  |
|  |  |  |  |  | | **3** |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | | | | | | *Leave blank* |
| **Question 17** | | | | |  |  |
| **1** | **7** | **.** | **1** | def PrintResult(A, B, NextPlayer):  print('Game ended')  print(NextPlayer, 'cannot make a move')  Ascore = A[0][0]-(countNumberOfPieces(A)+3\*(A[0][1]))  Bscore = B[0][0]-(countNumberOfPieces(B)+3\*(B[0][1]))  PrintPlayerPieces(A, B)  if Ascore>Bscore:  print("player B won")  elif Ascore==Bscore:  print("its a draw")  else:  print("palyer A won")  print("A's score: ",Ascore)  print("B's score: ",Bscore)  def countNumberOfPieces(Player):  count=0  for i in range(1,len(Player)):  if Player[i][0] != -1 and Player[i][1] != -1:  count+=1  return count | |  |
| **1** | **7** | **.** | **2** | A close up of text on a white background  Description automatically generated | |  |
|  |  |  |  |  | |  |
|  |  |  |  |  | | **10** |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | | | | | | *Leave blank* |
| **Question 18** | | | | |  |  |
| **1** | **8** | **.** | **1** | B | |  |
| **1** | **8** | **.** | **2** | def MoveDame(Board, Player,OpponentsPieces, NewRow, NewColumn,PlayersPieces):  valid = False  while not valid:  piece = input("choose which of your opponent's pieces you want to take ")  if piece[0].lower() == Player.lower():  DisplayErrorCode(6)  continue  try:  pieceNumber=int(piece[1:])  print(OpponentsPieces[pieceNumber][0],OpponentsPieces[pieceNumber][1])  if OpponentsPieces[pieceNumber][0]!=-1 and OpponentsPieces[pieceNumber][1]!= -1:  valid = True  NewRow = OpponentsPieces[pieceNumber][0]  NewColumn = OpponentsPieces[pieceNumber][1]  OpponentsPieces[pieceNumber][0] = -1  OpponentsPieces[pieceNumber][1] = -1  except:  DisplayErrorCode(5)  return NewRow, NewColumn    def MovePiece(Board, PlayersPieces, ChosenPiece, NewRow, NewColumn,OpponentsPieces):  Index = int(ChosenPiece[1:])  CurrentRow = PlayersPieces[Index][ROW]  CurrentColumn = PlayersPieces[Index][COLUMN]  Board[CurrentRow][CurrentColumn] = SPACE    if NewRow == BOARD\_SIZE - 1 and PlayersPieces[Index][DAME] == 0:  Player = 'a'  PlayersPieces[0][1] += 1  PlayersPieces[Index][DAME] = 1  ChosenPiece = ChosenPiece.upper()  NewRow, NewColumn = MoveDame(Board, Player,OpponentsPieces, NewRow, NewColumn,PlayersPieces)  elif NewRow == 0 and PlayersPieces[Index][DAME] == 0:  Player = 'b'  PlayersPieces[0][1] += 1  PlayersPieces[Index][DAME] = 1  ChosenPiece = ChosenPiece.upper()  NewRow, NewColumn = MoveDame(Board, Player,OpponentsPieces, NewRow, NewColumn,PlayersPieces)  PlayersPieces[Index][ROW] = NewRow  PlayersPieces[Index][COLUMN] = NewColumn  Board[NewRow][NewColumn] = ChosenPiece  return Board, PlayersPieces | |  |
| **1** | **8** | **.** | **3** | A screenshot of text  Description automatically generated | |  |
| **1** | **8** | **.** | **4** | A close up of text on a white background  Description automatically generated | |  |
|  |  |  |  |  | |  |
|  |  |  |  |  | | **12** |

**If you have made any separate screen captures, put a reference in the correct place in this Electronic Answer Document.**

**Ensure that your Centre Number, Candidate Name and Candidate Number are on each sheet.**

**Attach them to the back of this document.**