**Worksheet 1 Hardware and software Answers**

**Task 1 Disk Defragmentation**

1. Here is a simplified diagram showing the contents of a hard disk drive.

Over time, different programs have been installed, updated, removed and reinstalled.

1. A user installs a new graphics package (**GP**) that takes up 5 blocks and a screen capture utility (**SC**) that takes up 2 blocks. Add **GP** and **SC** to the diagram to show where the memory management system might store the new blocks.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| OS | OS | OS | OS | WP | WP | OS | **GP** |
| Music | Browser | **GP** | WP | WP | Data | **GP** | Data |
| **GP** | Data | **GP** | **SC** | **SC** | Data | SS | Browser |
| OS | Music | SS |  | SS | Browser |  | SS |

The positioning of GP and SC blocks is not important, but students need to have placed the correct number of each. It might make more sense to place SC together in contiguous blocks in order to avoid fragmenting one of the two programs.

1. Defragment the drive to put all of the programs and files together so that they fit into continuous segments. The OS has been done for you.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| OS | OS | OS | OS | OS | OS | WP | WP |
| WP | WP | **GP** | **GP** | **GP** | **GP** | **GP** | Music |
| Music | Browser | Browser | Browser | Data | Data | Data | Data |
| **SC** | **SC** | SS | SS | SS | SS |  |  |

NB: Specific order of programs/data is not important - the aim is to have the blocks constituting each file grouped together.

1. Explain why the defragmented hard disk drive may improve performance.

Because each program/file is now in one contiguous group of sectors, the next block to be accessed is physically next to the previous one. This reduces read/write time as the hard drive doesn’t have to spin round to the next sector somewhere else on the disk, and the read/write head does not need to move across the disk.

1. Explain why a certain amount of free hard disk drive space is needed to carry out disk defragmentation.

Some empty space is needed because the data needs to be moved out of the way in order to free up the space to put the data back in contiguous groups. (This is similar to a sliding picture puzzle where one corner is empty so you can slide the squares around.)

2. Windows automatically creates “System Restore” points at regular intervals, for example daily or weekly. Why might you need to use a System Restore utility? Do some research to find out more about this utility and how you run it. Write down some facts you discover.

 (Very open question but useful to know about this if you are a Windows user!)

3. Complete the table below, describing each type of systems software

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **Operating System** | **Utility** | **Library** | **Translator** |
| Disk defragmenter |  | ✓ |  |  |
| Microsoft Windows | ✓ |  |  |  |
| DLL file for web browsing |  |  | ✓ |  |
| Android software | ✓ |  |  |  |
| Backup manager |  | ✓ |  |  |
| Java Compiler |  |  |  | ✓ |
| Anti-virus program |  | ✓ |  |  |
| Python interpreter |  |  |  | ✓ |
| Module to generate random numbers |  |  | ✓ |  |

4. Other than those above, name and describe the purpose of **two** different utility programs

Firewall – to protect a computer or network from malicious access to data

Automatic updates –automate the process of downloading and installing software updates

Encryption / decryption – to hide the contents of data, to keep data secure

Compression / decompression – to reduce the size of files for storage/transmission, to group several files together

File restore – repair/restore corrupt or damaged files

System cleanup – remove unwanted, leftover or temporary files

(any other sensible answer)

# Task 2 Systems software

1. In a language which you are familiar with, write a program to compare the execution time of two algorithms in milliseconds. You will need to import a **library program** to do this.

 An example of a Python program is given below; you can use this one or a different one using algorithms you have already written.

 What does this program demonstrate?

 It demonstrates that one method of adding to a list is much faster than the other – this ties in with the “desirable properties of an algorithm”, that it should execute as fast as possible.

 #timing two methods of adding to a list.

#append and concatenate using +

import time # import a library program

def appendToList(n):

 alist = []

 t0 = time.clock()

 for i in range(n):

 alist.append(i)

 t1 = time.clock()

 runtime = round((t1 - t0)\* 1000,2)

 print("time to append",n, " items to list ",runtime, "milliseconds")

def concatenateList(n):

 alist = []

 t0 = time.clock()

 for i in range(n):

 alist = alist + [i]

 t1 = time.clock()

 runtime = round((t1 - t0)\* 1000,2)

 print("time to concatenate ",n, " items to list ",runtime, "milliseconds")

#main

k = int(input ("How many items do you want to add to your list? "))

appendToList(k)

concatenateList(k)

quit = input()