1. A video game can be described as a "soft real-time interactive agent-based temporal simulation" (Gregory, 2018, pp. 8-11). Explain and elaborate on the meaning of each part of the given definition. Discuss how this characterization corresponds to a general view of games. Are there any computer games that do not nicely fit this description? Explain why / why not.

Answer: The video game is modelled mathematically so that it can be manipulated by a computer to represent a simplification of reality, so it is considered a simulation.

It is agent-based as it has a number of distinct entities known as “agents” interact such as vehicles, characters, fireballs, power dots and so on.

It is interactive as it will respond to unpredictable inputs from its human player(s).

It is a temporal simulation as the game world is dynamic- in which the state of the game world changes over time as the game’s events and story unfold.

There are games such as turn-based games like chess or turn-based strategy games which does not fit into the above description, since the game could freeze for as long as one person does not make a move.

2. Give a C function that allocates memory from the free store (heap) for an array for n integers (of type int) where n is given as an argument to the function. The function returns the allocated memory (or NULL in case of error) to the caller.

Answer:

```
#include <stdio.h>

#include <stdlib.h>

int *f (int n){

  int *pointer = NULL;

  pointer = (int*)malloc(n*sizeof(int));

  return pointer;

}
```

The function above would return a pointer with size n, you can initialise a pointer with size n by writing int *pointer = f(10)

Then, give a program that uses the function to allocate space for an array of 10 integers, inserts the numbers from 1 to 10 into the array (in ascending order), prints out the 3rd element of the array, deallocates the memory, and finally terminates. Compile and try out your program.

Answer:

```
#include <stdio.h>

#include <stdlib.h>

int *f (int n){

  int *pointer = NULL;

  pointer = (int*)malloc(n*sizeof(int));

  for (int i = 0; i <10;i++){

    *(pointer+i) = i+1;

  }

  return pointer;

}


int main(void) {

  int *a = f(10);

  printf("%d",*(a+2));

  free(a);

  return 0;
```

}

3. Answer the following questions about the compilation of C programs. a. Explain the difference between a definition and a declaration in C programs.

AmswerDefinition of a variable would hold the information of where the variable gets stored while declaration of a variable would hold the information of the name of the variable, type of value it holds, and its initial value (if any).

What is indicated by a declaration that uses the extern modifier?

Answer: Declaration that uses the extern modifier means that we do not want to allocate memory to the variable.

What does the keyword "static" mean in C?

Answer: The keyword static would make the scope of the variable global and available throughout the program.

b. Explain how C (and C++, too) supports separate compilation. Define what a translation unit is. Explain how the #include directive works in C (and C++).

Answer: C or C++ support separate compilation, where pieces of the program can be compiled independently through the two stage approach of compilation and then linking, so changes to one class would not necessarily require the re-compilation of the other classes. The compiled pieces of code (.o or .obj files)[8] are combined through the use of the linker.

As C program needs not be translated at the same time, the text of the program is kept in units called source files, (or preprocessing files). A source file together with all the headers and source files included via the preprocessing directive #includeis known as a preprocessing translation unit. After preprocessing, a preprocessing translation unit is called a translation unit.

Also explain what header files are, and why they are needed in building programs.

Answer: Header file is a file with extension .h which contains C function declarations and macro definitions to be shared between several source files. It is needed because we can include the header file in our source file if we require its function prototypes.

Explain also, what goes on during linking (hint: look up "Linker (Computing)" from Wikipedia).

Answer: Linker is a program that helps to link a object modules of program into a single object file, it collects and maintain piece of code and data into a single file, and takes object modules from assembler as input and forms an executable file as output for loader.

4. Give a sample C program that uses multiple translation units (you could use the program in Exercise 2. as a basis, for example). If possible, compile and try out your program. Explain your code.

```
//main.c

#include “foo.c”

int main(void) {

  foo();

}

//foo.c

#include <stdio.h>

void foo()

{

    printf("Hello World!");

}
```

I have two files named main.c and foo.c, where foo.c has one function called foo(), my main function would just include the foo.c file and invoke the function call foo.c().