Please Note that for all questions on this exam you may assume that the BOOLEAN type is defined and has valid values of `TRUE` and `FALSE`.

Question 1 – Short answer Questions (5+5+5+5+5+5)

a) what is the output of the following program:

```
int i=4;
int doit(int i)
{
      i=5;
      return i*2;
}

int main(void)
{
      i=doit(i);
      printf("%d\n", i);
}
```

```
10
```

b) what is the output of the following program:

```
#include <stdio.h>
#include <stdlib.h>

int main(void)
{
     char * message[] =
     {
         "darkness", "cannot", "drive", "out", "darkness"
     };
     char * word = *(message + 3);
     puts(message[4]);
     putchar(*(*(message+4)+3)+6);
     puts(*(message+2)+3);
     puts("darkness"+2);
     puts(word);
     return EXIT_SUCCESS;
}
```

```
darkness
qve
rkness
out
-1 for each thing incorrect
```

c) The following is an error given by gcc – what could be wrong? :

```
/tmp/ccMmCq0c.o: In function `main':
1c.c:(.text+0x5): undefined reference to `doit'
collect2: error: ld returned 1 exit status
```

a function called doit() has been declared but not defined

d) the following is the definition of a macro to implement division of two numbers. There are a range of ways this macro could be used incorrectly. Provide a correct definition that does not have these problems:

```
#define DIV(A,B) A/B
```

```
#define DIV(A,B) ((A)/(B))
```

e) There is a problem with the **output** of this function (there is also a problem with the input but we will look at that in the next question). What is the problem and how could we fix this?

```
#include <stdio.h>
#include <stdlib.h>
#define LEN 80
#define EXTRACHARS 2
/* converts metres to kilometres */
#define M2KM 1000
int main(void)
{
    int metres, kilometres;
    char line[LEN + EXTRACHARS];
    printf("How many metres have you run this week? ");
    fgets(line, LEN + EXTRACHARS, stdin);
    metres = atoi(line);
    kilometres = metres/M2KM;
    printf("You have run %d kilometres this week\n", kilometres);
    return EXIT_SUCCESS;
}
```

kilometres is an integer – we have integer division and so it is not as accurate as using floating point division.

f) There were also several problems in the previous function with how **input** was **validated** (or in fact not validated). Identify at least one of those problems and provide a solution to that problem.

Three problems: we do not check for leftover input in the buffer. We do not check for ctrl-d. We do not validate that there is actually an int in the string. 2.5 for correctly identifying a problem. 2.5 for providing a valid solution.

Question 2 – Generics (10+5+15)

a) Write a function called iterate() to iterate over any type of array. The function should use the passed in function pointer to perform some operation on each element of the array.

```
void iterate(void * array, size_t element_size,
             size_t element_count, void alter(void*));
```

```
void iterate(void * array, size_t element_size,
       size_t element_count, void alter(void*))
{
/* 2 marks for variable declarations */
   void * base = array;
   unsigned count;
/* 2 marks for correct counting of the number of elements */
   for(count = 0; count < element_count; ++count)
   {
/* 3 marks for correct dereference and calling of function using function pointer */
      alter(base);
/* 3 marks for the correct incrementing of the pointer */
      base = ((char*)base) + element_size;
   }
}
```

b) Write two functions to be passed into the function you defined in the previous question. One should add 5 onto an integer, the other should add 5 onto a double. You should perform any necessary casts to achieve this task. The function prototypes are:

```
void add_5_int(void* num);
void add_5_dbl(void* num);
```

2.5 marks for correct implementations of functions to alter the data.

```
void add_5_int(void* num)
{
   *((int*)num)+=5;
}

void add_5_dbl(void* num)
{
   *((double*)num)+=5;
}
```

c) Write a main function where you create two arrays – one of type double and one of type int and initialise each one with ten values (it doesn't matter what they are). Use the functions above to process these arrays – so call iterate() and pass in the appropriate function pointer. Next, print out the altered values in these arrays.

```
int main(void)
{
/* 3 marks for correct variable declarations */
   int int_array[] = {1,3,5,7,9,11,13,15,17,19};
   double dbl_array[] = {.1,.3,.5,.7,.9,1.1,1.3,1.5,1.7,1.9};
   unsigned count;

/*4 marks for each call to iterate */
   iterate(int_array, sizeof(int), 10, add_5_int);
   iterate(dbl_array, sizeof(double), 10, add_5_dbl);
/* 2 marks for each loop to print out the data */
   for(count = 0; count < 10; ++count)
   {
      printf("%d\t", int_array[count]);
   }

   printf("\n");
   for(count = 0; count < 10; ++count)
   {
      printf("%f\t", dbl_array[count]);
   }
   return EXIT_SUCCESS;
}
```

Question 3 – Dynamic Memory Allocation(5 + 25 + 10)

a) Write a function that creates a new image of the width and height specified (including allocating a 2d array pixels using malloc()) and initializes the red, green and blue elements of each pixel to 0. If there are any problems allocating memory, you should return `NULL` from this function. The function prototype for this function is:

```
struct image * create_image(unsigned width, unsigned height)
{
/* declare variables - 1 mark */
    unsigned count;
/* allocate memory and handle errors - 1.5 marks */
    struct image * newim = malloc(sizeof(struct image));
    if(!newim) return NULL;
    newim->pixels = malloc(sizeof(struct pixel*) * width);
    if(!newim->pixels) return NULL;
/* 2 marks for correct initialisation of memory -allocate
memory of each row and set it all to 0.*/
    for(count = 0; count < width; ++count)
    {
        newim->pixels[count] = malloc(sizeof(struct pixel) *
height);
        if(!newim->pixels[count]) return NULL;
        memset(newim->pixels[count], 0, sizeof(struct pixel) *
height);
    }
/* 0.5 - assign width and height */
    newim->width = width;
    newim->height=height;
    return newim;
}
```

b) We want to be able to load in an 'image' from file. Now, instead of using a binary format that you are not familiar with, we will assume the following format for a text file that represents an image. The first line of the text file will have two comma separated values: the width and the height of the image. Each other line will have three comma separated values to represent the red, green and blue of each pixel respectively.

Write a function to load in such an image file. You should validate that there are two comma separated numbers on the first line and three comma separated numbers on every other line. You should also validate that all numbers to represent colors are between 0 and 255 and there are not more lines to represent pixels than was specified via the width and height at the start of the image. The pixel values should be assigned going from left to right across a row and then moving to the next row.

As part of this function you should call create_image() defined above to allocate space for the image to be loaded. You may assume that the infile parameter is valid and refers to a file that has just been opened prior to being passed in here. Your function should return the struct image pointer created and assigned. The function prototype is:

struct image * load_image(FILE * fp);

```
struct image * load_image(FILE * infile)
    /* 20 minutes */
{
/* 3 marks for variable declarations */
    struct image * newim = NULL;
    char line[LINELEN + EXTRACHARS];
    /* lots of tokenization */
    char * tok;
    char * end;
    unsigned width, height;
    unsigned num_pixels=0;
    unsigned pixel_count=0;
    struct pixel pix;

    /* get the width and height - 4 marks*/
    if(fgets(line, LINELEN + EXTRACHARS, infile) == NULL)
        return NULL;
    tok = strtok(line, DELIMS);
    if(!tok) return NULL;
    width = (unsigned)strtol(tok, &end, 0);
    if(*end)
        return NULL;
    tok = strtok(NULL, DELIMS);
    height = (unsigned)strtol(tok, &end, 0);
    if(*end)
        return NULL;
    tok = strtok(NULL, DELIMS);
    if(tok)
        return NULL;
    num_pixels = width * height;
/* correct call to create_image() - 2 marks */
    newim = create_image(width, height);
/* retrieve and each line - 4 marks */
    while(fgets(line, LINELEN + EXTRACHARS, infile))
    {
        if(line[strlen(line)-1] != '\n')
            return NULL;
/* tokenise - 4 marks */
        tok =strtok(line, DELIMS);
        if(!tok) return NULL;
/* convert ints - 4 marks*/
        pix.red = (char) strtol(tok, &end, 0);
        if(*end)
            return NULL;
        tok=strtok(NULL, DELIMS);
        if(!tok)
            return NULL;
        pix.green = (char)strtol(tok, &end, 0);
        if(*end)
            return NULL;
        tok = strtok(NULL, DELIMS);
        if(!tok)
            return NULL;
        pix.blue = (char) strtol(tok, &end, 0);
        tok = strtok(NULL, DELIMS);
        if(tok)
            return NULL;
/* insert data into array - 4 marks */
        newim->pixels[pixel_count %width][pixel_count/width] =
pix;
        ++pixel_count;
    }
    if(num_pixels != pixel_count)
        return NULL;
    return newim;

}
```

c) Write a main function that takes in as a command line argument the name of a file to open that is assumed to be in the format specified above. You should print out each row of red, green and blue values making it clear when a new row begins. You should also free any memory allocated and close the file that was opened by this function.

```
/* 1 mark for correct main prototype */
int main(int argc, char * argv[])
{
/* 2 marks for variable declarations */
  struct image * image = NULL;
  FILE * fp;
  unsigned width,height;
  struct pixel  pix;

/* 2 marks for opening file and verifying it */
  fp = fopen(argv[1], "r");
  if(!fp) return EXIT_FAILURE;
  image = load_image(fp);
/*2 marks for calling load_image and checking that the file loaded correctly */
  if(!image)
  {
    printf("image is NULL");
    return EXIT_FAILURE;
  }
/* 2 marks for printing the values */
  for(height=0; height < image->height; ++height)
  {
    printf("row %d\n", height);
    for(width=0; width< image->width; ++width)
    {
      pix = image->pixels[width][height];
      printf("red: %d, green: %d, blue: %d\n",
        pix.red, pix.green, pix.blue);
    }
    printf("end row\n\n");
  }
/* 1 mark for call to fclose() */
  return EXIT_SUCCESS;
}
```

Question 4 – Software Design (10+10+5+5)
Consider a program to manage the collection of research papers that a student might read in preparation for writing a thesis. Each paper needs to store the following details (we'll stick to just academic papers as books and other source materials might need other data):

- Article title:
- Year published:
- Journal Name:
- Volume no:
- Issue no:
- Pages used:

We want to be able to load a text file with entries for each paper, edit the details about a paper, add papers, delete papers and save these papers back to disk.

a) make a list of at least 5 modules (.c and .h pairs) that you would create for this program. for each module, write down a sentence or two explaining its purpose and what would be contained in that module.

For this question there should be 5 modules, eg:
Io.c/h
Main.c/h
paper.c/h
paperlist.c/h
interface.c/h.

The filenames should relate to what they will hold and there should be a meaningful comment that explains this. 1 mark for each listed correct module and 1 mark for each correct description.

b) One of the modules created in the previous question would have been for managing a paper. Write down all the contents of the header file for that module.

Paper.c

3 marks for #includes / #defines
4 marks for the datastructure for a paper
3 marks for reasonable function prototypes

c) How can we limit the visibility of a function so that it can only be run by other functions in the same file? Provide an example and explanation of what you are doing.
We need the function to be of static storage class.
Eg, in interface.c we might have validation functions that will only be callable within that module. Say, BOOLEAN enter_paper(struct paperlist * list) might call functions like validate_pages(int i) higher up in the file it would be declared as:

```
static BOOLEAN validate_pages(int i)
{
}
```

But would only be callable later in that file and would not have the prototype in interface.h

d) What is the purpose of the auto storage class? Does it make sense to ever explicitly use the auto keyword? Explain your answer.

The auto keyword means that storage is automatically allocated on the stack when a variable is declared. We don't bother using it in practice as it is the default storage class for local variables.

Question 5 – Linked Lists (10+15+30)

Perry's Pretty Pastries is the name of a pie maker that sells pies direct to the public in the Central Business District of Melbourne.

Perry has asked you to construct a program for his company that will allow users to enter the products they would like to purchase from a preloaded list of products. Perry does not know much about these new-fangled computer programs but he has been told that a linked list would be a good way to store this data. Perry also wants to be able to store the total number of each type of pie that has been sold recently as well as the overall number of pies sold.

a) Provide a set of struct definitions that will allow Perry's requirements to be satisfied. You should pay attention in particular to good software design and the development of a modular solution. The only restriction is that the handle that refers to the list should be called pie_list.

```
/* 3 marks for pie data */
struct price
{
   unsigned dollars, cents;
};

struct pie
{
   struct price price;
   char name[NAMELEN + 1];
   char flavour[FLAVLEN + 1];
   unsigned num_sold;
};

/* 3 marks for links */
struct pie_node
{
   struct pie_node * next;
   struct pie * data;
};
/* 3 marks for header structure */
struct pie_list
{
   struct pie_node * head;
   unsigned total_pies;
   unsigned pies_sold;
};
/* 1 mark for typedef */
```

typedef struct pie_list pie_list;

b) Periodically, Perry wants the ability to reset the number of each type of pie sold to 0 as well as the global total of pies sold. Please write this function. The function prototype is:
`void reset_number_sold(pie_list * list);`

```
void reset_number_sold(pie_list * list)
{
/* 3 marks for variable declarations */
    struct pie_node *current;
    if(list->head == NULL)
        return;
/* 5 marks for correct initialisation and loop condition */
    current = list -> head;
    while(current != NULL)
    {
/* 3 marks for correctly resetting number sold */
        current->data->num_sold=0;
/* 3 marks for correct iteration to next node */
        current = current -> next;
    }
/* 1 mark for setting the global count of sold */
list->total_sold=0;
}
```

c) For reasons of reporting to the shareholders, Perry may want to sort the pie list according to the number sold with the highest selling pie at the top. Please implement this function. The function prototype is:

```
void sort_by_number_sold(pie_list * list)
{
/* 3 marks for variable declarations */
    pie_list newlist;
    struct pie_node * tail;
    tail = newlist.head = NULL;

    if(list->head == NULL)
        return;
    while(list->head != NULL)
/* 10 marks for correct outer loop */
    {
        struct pie_node * largest;
        struct pie_node * current, * prev=NULL;
        struct pie_node * prevlargest;

        prevlargest = NULL;
        current = list->head;
        largest = current;

/* 10 marks for correct inner loop */
        while(current != NULL)
            /* search for largest */
        {
            while(current != NULL)
            {
                if(current->data->num_sold > largest->data-
>num_sold)
                {
                    prevlargest = prev;
                    largest = current;
                }
                prev = current;
                current = current -> next;
            }
        }
/* 5 marks for correct removal and inseration into sorted list
*/
        if(prevlargest)
        {
            prevlargest->next = largest->next;
        }
        else
        {
            list->head = list->head->next;
        }
        largest->next = NULL;
        if(tail)
        {
            tail->next = largest;
            tail = largest;
        }
        else
        {
            newlist.head = largest;
            tail = newlist.head;
        }
    }
/* 2 marks for correct resetting of the head */
    list->head = newlist.head;
}
```