VICTORIA UNIVERSITY OF WELLINGTON
TE HERENGA WAKA

EXAMINATIONS – 2023

TRIMESTER 1

NWEN 241
Systems Programming
21 June 2023

Time Allowed: TWO HOURS

CLOSED BOOK (SELECTED MATERIALS ONLY)

Permitted materials: Only silent non-programmable calculators or silent programmable calculators with their memories cleared are permitted in this examination.

Paper foreign to English language dictionaries are allowed.

Electronic dictionaries are NOT allowed.

NO other material is allowed.

Instructions: Attempt all **SEVEN** questions.

The questions are grouped into two main sections:

- SECTION A: C Pogramming [40 Marks]
- SECTION B: Process Management and C++ Programming [80 marks]

All answers must be written in the boxes provided in this test paper.

The examination consists of **120** marks in total.

Within each question, the marks for subparts are shown.

SECTION A C programming (40 marks)

1. **Arrays, Strings and Pointers** **(15 marks)**

   (a) Fill in the blanks (i) and (ii) such that the following C program would be able to traverse the entire array arr. **(4 marks)**

```
 #include <stdio.h>

int main(void) {
   int arr[10] = {1 2,3,4,5};

   for (int * p = _______ /* (i) */ ; p < ________ /* (ii) */ ; p++)
      printf("%d ", * p);

   return 0;
}
```

   Answer for (i):

```
arr or &arr or &arr[0]
```

   Answer for (ii):

```
arr+10 or arr+sizeof(arr)/sizeof(int)
arr can be replaced by either &arr or &arr[0]
```

   (b) Consider the following C program: **(2 marks)**

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

#define HELLO "Hello"
#define WORLD "World"

int main(void) {
   char msg[] = HELLO WORLD;
   printf("%s %d", msg, strlen(msg));
}
```

Will the program compile, and if so, what is its output?

The program will compile. Its out is `HelloWorld 10`.

(c) Fill in the blanks (i)–(iv) such that the following function would return the number of occurrences of letter the 'a' (both upper and lower case) in the input `str` which is a null-terminated string. (You are **not** allowed to declare any other variable.) **(4 marks)**

```
 int count_As(char * str) {
   int count = 0;
   while (________________ /* (i) */ ) {
      if ( * str == 'A' || * str == 'a')
         _____________; /* (ii) */
      ______________; /* (iii) */
   }
   __________________; /* (iv) */
}
```

Answer for (i):

`*str != '\0'` or `!*str`

Answer for (ii):

`count++` or `++count`

Answer for (iii):

`str++` or `++str`

Answer for (iv):

`return count`

(d) Given the following variable declarations:

```
short a[] = {2, 4, 8, 16, 32, 64};
short *p = a;
short **pp = &p;
```

Suppose that a `short` occupies 2 bytes in memory. The array `a` is at memory address 100, while `p` is at memory address 200 (all addresses are in decimal).

i. What is the numeric value of the expression `p`? **(1 mark)**

100

ii. What is the numeric value of the expression `*p+1`? **(1 mark)**

2+1 = 3

iii. True or False: `&a[2] == p+2` **(1 mark)**

True

iv. What is the numeric value of the expression `pp`? **(1 mark)**

200

v. Using the pointer to pointer `pp`, write an expression that accesses the second element of array `a`, that is, `a[1]`. **(1 mark)**

`*(*pp+1)`

2. **Structures & Dynamic Memory** **(10 marks)**

(a) Consider the following C structure, with tag `point3d` and consisting of 3 float members `x`, `y` and `z`: **(2 marks)**

```
 struct point3d {
   float x;
   float y;
   float z;
};
```

Use typedef to define a new type `point3d_t` from the above structure.

```
typedef struct point3d point3d_t;
```

(b) Consider the following declaration where `struct point3d` is the structure defined in (a): **(2 marks)**

```
 struct point3d *p;
```

Write a C statement using `calloc()` that will dynamically allocate an array of 15 `struct point3d` elements, and let `p` point to that memory.

```
p = (struct point3d *) calloc(15, sizeof(struct
point3d));
```
The typecasting of the return value is optional.

(c) Implement the function with prototype **(4 marks)**

```
 struct point3d *new_point3d(float x_, float y_, float z_);
```

that will dynamically allocate memory for a `struct point3d *` using `malloc()`, initialize the members `x`, `y` and `z` to `x_`, `y_` and `z_`, respectively, and return a pointer to the allocated memory. If the memory allocation fails, the function should return NULL.

```
struct point3d *new_point3d(float x_, float y_,
float z_)
{
    struct point3d *p = malloc (sizeof(struct
point3d));
    if(p == NULL) return NULL;
    p->x = x_;
    p->y = y_;
    p->z = z_;
    return p;
}
```

(d) Write a short C code snippet to illustrate the problem of memory leak. **(2 marks)**

The code should show the following:
(1) A pointer is first used to point to a dynamically allocated memory;
(2) The same pointer is then reassigned to point to another memory block (even NULL) without freeing the block in (1).

3. **File Stream I/O and Command Line Arguments** **(15 marks)**

Study the following C program which contains blanks.

```
#include <stdio.h>

#define INPUT_FILE "input.txt"

int main(int argc, char * argv[]) {
   FILE * infp, * outfp = NULL;
   int c, d;

   if (argc == 1)
      _______________________; /* question (a) */
   else if (argc == 2)
      _______________________; /* question (b) */

   if (infp == NULL)
      return 0;

   while (____________ /* question (c) */ ) {
      ____________________; /* question (d) */
      d = toupper(c);
      if (outfp != NULL)
         _____________________; /* question (e) */
   }

   if (outfp != NULL)
      fclose(outfp);
   fclose(infp);

   return 0;
}
```

(a) Write a single C statement to open file `INPUT_FILE` for reading, and assign the opened file stream to `infp`. **(2 marks)**

```
infp = fopen(INPUT_FILE, "r")
```

(b) Write a single C statement to open the file passed as the first command line argument for writing, and assign the opened file stream to `outfp`. **(2 marks)**

```
outfp = fopen(argv[1], "w")
```

(c) Write an expression using `feof()` that tests the end-of-file of `infp` is not yet reached. **(2 marks)**

```
!feof(infp)
```

(d) Write a single C statement that reads a single character from the stream `infp` and stores it in `c`. **(2 marks)**

Either
```
c = fgetc(infp)
```
or
```
fscanf(infp, "%c", &c)
```

(e) Write a single C statement that outputs `d` as a single character to the stream `outfp`. **(2 marks)**

Either
```
fputc(d, outfp)
```
or
```
fprintf(outfp, "%c", d)
```

(f) Suppose that you have completed the program correctly and compiled the code to a binary executable file named `execfile`. Suppose further that you executed the file under a directory where a file named `input.txt` exists which contains

```
Hello world
```

i. If the command you entered in the terminal was **(2 marks)**

```
./execfile hello.txt
```

what would be the contents of `hello.txt`?

```
HELLO WORLD
```

ii. If the command you entered in the terminal was **(3 marks)**

```
./execfile input.txt
```

what would be the contents of `input.txt` after execution? Briefly explain your answer.

The contents would remain the same because the call to open `input.txt` would fail and `outfp` would therefore remain NULL.

SECTION B Process Management and C++ programming (80 marks)

4. **Process Management** **(15 marks)**

(a) How many times will the following C program print `NWEN241`? **(3 marks)**

```
#include<stdio.h>
#include<sys/types.h>
#include<unistd.h>

int main() {
   fork() && fork();
   fork();
   fork();
   printf("NWEN241\n");
}
```

12

(b) Name the process that owns the process ID 1 in Linux operating system. **(2 marks)**

init

(c) Fill in the blank. Fork returns ________ to the parent process on success. **(2 marks)**

process id of the child

(d) Which of the following system calls used in Socket programming is a blocking system call? **(2 marks)**

(a) `accept`
(b) `socket`
(c) `bind`
(d) `listen`

(a) or accept

(e) In which of the following system calls an identical copy of the original process is created? **(2 marks)**

(a) `execl`
(b) `fork`
(c) `accept`
(d) `wait`

(b) or fork

(f) You are given the following C program.

```
1. #include <stdio.h>
2. #include <stdlib.h>
3. #include <unistd.h>
4. #include <sys/wait.h>
5.
6. int gvar = 6;
7.
8. int main(void) {
9.    int lvar = 3;
10.   pid_t pid;
11.
12.   if ((pid = fork()) < 0) {
13.      printf("fork error\n");
14.   }
15.   if (pid == 0) {
16.      gvar++;
17.      lvar++;
18.   } else {
19.      wait(NULL);
20.   }
21.   printf("%ld %d %d\n", (long) getpid(), gvar, lvar);
22.   exit(0);
23. }
```

i. Which part of the program code is relevant to the child process created by the `fork` system call at line number 12. **(2 marks)**

`Line number :  15 - 17 and 21,22`

ii. Assume that the `fork` is successful and that the parent process ID is 12345 while the child process ID is 12346. What is the output of the program? **(2 marks)**

```
12346 7 4
12345 6 3
```

5. **C++ Classes** **(30 marks)**

(a) Consider the following code segment. **(2 marks)**

```
1. #include<iostream>
2.
3. class A {
4.   private: int item = 0;
5.   char icode = 'A';
6.   public:
7.     A():item(0),icode('
8.    A '){}
9.    A(int x, char c): item(x),icode(c) {}
10.   A(A & a): item(a.item),icode(a.icode) {}
11. };
12.
13.int main() {
14.  A a1(1, 'A');
15.  A a2 = a1;
16.  std::cout<<"two objects created";
17.  return 0;
18. }
```

Which constructor type is invoked at line number 15?

(a) Default constructor
(b) Parameterized constructor
(c) Copy constructor
(d) No constructor is invoked

(c) or Copy constructor

(b) State True or False. Declaring a constructor as `explicit` prevents implicit casting of constructor arguments to class objects. **(1 mark)**

True

(c) State True or False. In C++ the statement `delete p;` deallocates the memory pointed-to by the variable `p` and not the variable `p`. **(1 mark)**

True

(d) Consider the following code segment. Write the statement to access the variable `a` of the namespace `Box1`. **(2 marks)**

```
  namespace Box1 {
  int a = 4;
}

namespace Box2 {
  int a = 12;
}
```

`Box1::a`

(e) Consider the following code segment. In which order are the constructors invoked when an object `c` of class `C` is created? **(2 marks)**

```
  class A {
   int a;
   public:
     A() {
       std::cout << "A";
     }
 };

 class B {
   int b;
   public:
     B() {
       std::cout << "B";
     }
 };

 class C: public B, public A {
   int c;
   public:
     C() {
       std::cout << "C";
     }
 };
```

B A C

(f) Consider the following code snippet.

```
#include<iostream>

class A {
  public: int a;
  void f1(void) {
    a = 10;
  }

  void f2(void) {
    aa = 20;
  }

  int f3(void) const {
    f1();
    f2();
    return 0;
  }
  protected: int aa;
  private: int aaa;
};

int main() {
  A objA;
  objA.f3();
  return 0;
}
```

For an object `objA` of class `A`, state whether the following statements are True or False.

i. The statement `ObjA.aa` = 10; is valid **(2 marks)**

False

ii. The statement `ObjA.aaa = 10`; is valid. **(2 marks)**

False

iii. The statement `ObjA.a` = 10 is valid. **(2 marks)**

True

iv. It is not possible to instantiate `ObjA` as there is no constructor **(2 marks)**

False

v. Calling the member functions `f1()` and `f2()` from within member function `f3()` is valid. **(2 marks)**

False

(g) Define a class `Cube` with the following members: **(4 marks)**

- a private integer member `side`
- a public default constructor that initializes the data member `side` to 1
- a public parameterized constructor that takes in an integer to initialize `side` using initializer list
- a destructor that prints a "Destructor invoked" message when an object of `Cube` gets destroyed
- a public method `volume` that returns volume of the `Cube` object for which it is invoked.

```
class Cube {
private: int side;

public: Cube() {
  side = 1;
}

Cube(int i): side(i) {}

int volume() {
  return side * side * side;
}
```

```
  ~Cube() {
    std::cout << "Destructor invoked";
  }
};
```

(h) List one difference between a friend function and a member function of a class. **(2 marks)**

(Any one)
Friend function is a non-member function that has access to private and protected members of a class. It is not in the scope of the class in which it is declared. While, a member function is in scope of the class in which it is declared.

A friend function cannot be called using object of the class. While a member function is called using object of the class.

A friend function can be declared in private, public or protected scope of the class without any effect. While, a member function can be declared in private, public or protected scope of the class but have scope accordingly.

(i) Given a class `Student`, which form of the operator `delete` would you use to deallocate the memory allocated by this statement. **(2 marks)**

```
Student *student_list = new Student[10]
```

(a) `delete student_list`
(b) `delete []student_list`
(c) `delete [10]student_list`
(d) `delete student_list[10]`

b

(j) What is the difference between the following two statements. **(2 marks)**

```
int* p = new int[5];
int* p = new int(5);
```

The first statement allocates memory to store 5 contiguous integers (an array of 5 integers) and stores begining address of this array in pointer p.

The second statement allocates memory that can store one integer, initializes it with value 5 and allocates the address of this memory to p.

(k) What will be the output of the following program. **(2 marks)**

```
#include <iostream>
using namespace std;

class Myclass {
  int x;
  public "
  Myclass() {
    x = 5;
  }
};

int main() {
  Myclass * c = new Myclass;
  cout << c -> x;
}
```

This program generates a compile time error - x is private within the context.

6. **Templates and Containers** **(25 marks)**

   (a) What will be the output of the following C++ program. **(5 marks)**

```
#include <iostream>
using namespace std;

template < typename T >
  void display(const T & x) {
    static int count = 0;
    cout << "x = " << x << " count = " << count << endl;
    ++count;
    return;
  }

int main() {
  display < int > (1);
  display < int > (1);
  display < double > (1.1);
  return 0;
}
```

```
x = 1 count = 0
x= 1 count = 1
x= 1.1 count = 0
```

   (b) What will be the output of the following C++ program. **(4 marks)**

```
#include <iostream>
using namespace std;

template < typename T >
  T max(T & p, T & q) {
    return (p > q ? p : q);
  }
int main() {
  int x = 155, y = 60, m;
  long a = 105, b = 59, n;
  m = max(x, y);
  n = max(a, b);
  cout << m << endl;
  cout << n << endl;
  return 0;
}
```

```
155
105
```

(c) What will be the output of the following C++ program. **(4 marks)**

```
 #include <iostream>

#include <list>

#include <iterator>

using namespace std;

void showlist(list < int > l) {
  list < int > ::iterator it;
  for (it = l.begin(); it != l.end(); ++it)
    cout << * it << " ";
  cout << "\n";
}
int main() {
  list < int > list1;
  for (int i = 1; i < 10; i = i + 2) {
    list1.push_back(i);
  }
  list1.pop_front();
  list1.reverse();
  showlist(list1);
  return 0;
}
```

9 7 5 3

(d) Give three advantages of using the generic vector class over a C-style array. **(3 marks)**

(Any three)There is a single definition of the vector container, but it can be used to define many different kinds (data types) of vectors.

Vectors are similar to dynamic arrays with the ability to resize automatically when an element is inserted or deleted. Their storage is handled automatically by the container.

Vectors also have safety features that make them easier to use than arrays, automated bounds checking and memory management.

Vector elements are placed in contiguous storage so that they can be accessed and traversed using iterators.

(e) Name two functions used by `iterators` to support traversing of containers in C++. **(2 marks)**

`begin()` and `end()`

(f) There are two possible types of access that a container can support: linear and random. What type of access does the following two containers provide? **(3 marks)**

(a) vector
(b) list

`vector` - random. `list` - linear

(g) What will be the output of the following program? **(4 marks)**

```
#include <iostream>
#include <vector>
using namespace std;

int main() {
  vector < int > myvector;
  myvector.push_back(78);
  myvector.push_back(16);
  myvector.front() += myvector.back();
  cout << myvector.front() << '\n';
  return 0;
}
```

94

7. **File Handling** **(10 marks)**

(a) Which of the following is a correct statement to read a maximum of 20 characters into a C-string s until the extracted character is 't'? **(2 marks)**

(a) `std::cin.getline(s, 20, 't');`
(b) `std::cin.getline(s, 21, 't');`
(c) `std::cin.get(s, 20, 't');`
(d) `std::cin.get(s, 21, 't');`

c

(a) ii only
(b) iii only
(c) ii and iv only
(d) i and iii only

(b) In C++, which of the following will read an entire line from keyboard and store it in a std::string variable str? **(2 marks)**

(a) `std::cin >> str;`
(b) `std::cin << str;`
(c) `std::getline(std::cin, str);`
(d) `std::gets(std::cin, str)`

c

(c) Give one difference between `std::get()` and `std::getline()` functions. **(2 marks)**

The `getline()` function reads a whole line, and uses the newline character transmitted by the Enter key to mark the end of input. The `get()` function is much like `getline()` but rather than reading and discarding the newline character, `get()` leaves that character in the input queue.

(d) Write a C++ code that will declare and open a text file records.csv for input. **(2 marks)**

```
ifstream ifs;
ifs.open(‘‘records.csv’’,‘‘r’’);
```

(e) Which of the following is not used as a file opening mode? **(2 marks)**

(a) `ios::trunc`
(b) `ios::binary`
(c) `ios::in`
(d) `ios::ate`

```
ios::trunc
```

* * * * * * * * * * * * * * *