UNIVERSITY OF BIRMINGHAM

School of Computer Science

Second Year Undergraduate

06-30203

30203 LI Systems Programming in C and C++

Main Summer Examinations 2022

[Answer all questions]

Exam paper

Question 1

The code below defines a singly-linked list. This consists of the following definition of node:

```
1  struct node {
2    int val;
3    struct node * next;
4  }
```

Also, it uses the following insert function:

```
5  struct node * insert(struct node * first, const int val) {
6    struct node * pred = NULL, * succ = first;
7    while (succ != NULL) {
8      if (succ->val < val)
9        pred = succ;
10     if (succ->val > val)
11       break;
12     succ = succ->next;
13   }
14   struct node * this = malloc(sizeof(struct node));
15   this->val = val;
16   this->next = succ;
17   if (pred != NULL) {
18     pred->next = this;
19     return first;
20   } else {
21     return this;
22   }
23 }
```

Note that the code above compiles correctly.

(a) Give an implementation in C for a function that prints all values in the singly-linked starting from the `first` node and following the `next` node one by one along the list. Use the following signature:

```
void print(struct node * first);
```

[3 marks]

(b) Give a function in C that frees all nodes in the list. Use the following signature:

```
void destroy(struct node * first);
```

[3 marks]

(c) Refer to the insert function above. Briefly describe what the sequence of printed value will be, with respect to any sequence of inserted values. Note that values are supposedly inserted as follows:

```
struct node * list = NULL;
while (<there is a value to insert>)
  list = insert(list, <next value to insert>);
```

[7 marks]

(d) Consider a program that inserts a sequence of values, prints the list, and destroys the list. Assume that print and destroy work as intended. The given insert function causes a memory leak. Identity where it is and how it occurs. Correct the insert function to remove the leak. Your solution should not alter the elements that are inserted and their order

[7 marks]

Question 2

(a) The question is about Main and Virtual Memory. Provide a brief answer.

(i) Why does a processor have a set of registers in addition to a large main memory? **[1 mark]**

(ii) A long-term scheduler controls the degree of multi-programming in an Operating System. The long-term scheduler can send a process to which state(s)? **[1 mark]**

(iii) Does adding more frames during Page Replacement always lead to improved performance? **[1 mark]**

(iv) A system is running with following measure behaviour: CPU utilization 10%; Paging disk 95% Other I/O devices 3%. Explain which of the following actions will improve CPU utilization and why?

i. Install more main memory

ii. Install a faster disk

iii. Changing the degree of multi programming **[3 marks]**

(b) Briefly describe what the possible consequences are of a buffer overflow in the kernel. **[6 marks]**

(c) Consider the following piece of kernel code. The intention is that whenever data is written to a proc-file, this data is written to a device. The device provides two functions: **start_transfer** starts the transfer of count bytes to the device and returns immediately, and **transfer_finished**, which is called by the device when the data transfer is finished. The function **kernelWrite** should return the number of bytes transferred to the device.

```
int total_transferred = 0;
/* total number of bytes transferred since module loaded */
int transferred = 0;
/* bytes transferred to device in single transaction */

/* called by device when transfer finished */
/* called in interrupt mode */
void transfer_finished(int count) {
  transferred = transferred + count;
  /* wakeup waiting process */
}

/* called every time data is transferred to kern
     ,as a result of writing to proc-file  */
int kernelWrite(char * buffer, int count) {
  /* buffer is pointer to user space */
  start_transfer(buffer, count);
  /* go to sleep until woken up in transfer finish */
  transferred = transferred + count;
  return transferred;
}

void init module(void) {
  /* set up proc-structure - code omitted */
  proc->write_proc = kernelWrite;
}
}
```

This kernel code compiles correctly but does not work as intended. Identify these errors and suggest remedies. If you think critical sections are required, it is sufficient to indicate begin and end of a critical section, and whether you would use semaphores or spinlocks to protect the critical section.

[8 marks]

Question 3

(a) Four processes running on a single-core processor (Table 1). Among all the Scheduling Algorithms, briefly explain which one you will prefer and which one you would like to avoid for the given scenario? **[5 marks]**

Table 1: List of Processes

Process	Type	Arrival time	Burst time
P1	CPU Bound	0	50
P2	I/O Bound	1	3
P3	I/O Bound	2	4
P4	CPU Bound	3	20

(b) Predict all possible outputs that the following C program will print to the console and briefly explain your answer. What will be the state of parent process? Briefly explain the behaviour of the program if we comment out the line number 16.

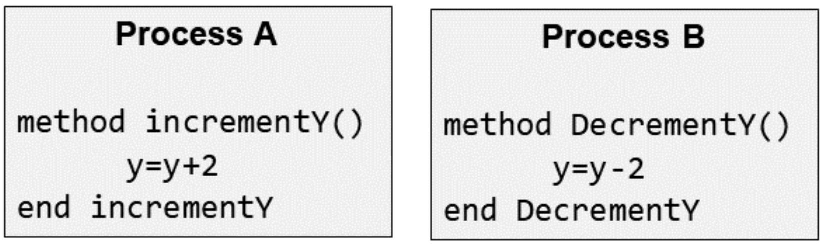
```
 1 #include <sys/types.h>
 2 #include <sys/wait.h>
 3 #include <unistd.h>
 4 #include <stdio.h>
 5 #include <stdlib.h>
 6 int main() {
 7    pid_t   wpid, child_pid;
 8    int status = 0;
 9    int i;
10    for(i = 0; i < 2; i++) {
11        if ((child_pid = fork()) == 0) {
12           printf("process \n");
13           exit(0);
14        }
15    }
16    while ((wpid = wait(&status)) > 0);
17    return 0;
18 }
```

[5 marks]

(c) Your computer system uses Round Robin scheduler and is not very responsive and so you decide to change the scheduling time quantum from 50 msec to 1 msec. Now the performance is even worse. Why is this happening? **[4 marks]**

(d) Consider a concurrent system with two processes A and B (Figure 1). Assume y is a shared variable with value of 5. Describe how a race condition is possible and provide a solution to prevent the race condition from occurring. **[6 marks]**

Figure 1: Concurrent processes A and B



Process A

```
method incrementY()
       y=y+2
end incrementY
```

Process B

```
method DecrementY()
       y=y-2
end DecrementY
```