Index number: .....

UNIVERSITY OF RUHUNA

Faculty of Engineering

End-Semester 6 Examination in Engineering: January 2022

Module Number: EE6206

Module Name: Operating Systems and Programming (N/C) –
Part I (MCQ)

[Forty Minutes only]

[This paper consists of 20 MCQs. Answer all questions, each question carries 0.5 marks.
Part I Will be collected at the end of forty minutes.
Write the index number in all pages.
Select only one correct answer for each of the questions.
Underline or cross the correct answer for each question.]

Q1 Which of the following is not a way of creating an empty text file **Doc1.txt**?

(i) Type `cat > Doc1.txt` in console. Then, press Enter. Then, press `Ctrl + D`.
(ii) Type `gedit Doc1.txt in console`. Then, press `Ctrl + S`.
(iii) Type `touch Doc1.txt` in console.
(iv) Type `gedit Doc1.txt in console`. Then, press `Ctrl + C`.
(v) Type `cat > Doc1.txt` in console. Then, press Enter. Then, press `Ctrl + C`.

Q2 What is the correct Linux console command to append the content of **doc1.txt** to an existing **doc2.txt**?

(i) `cat doc1.txt doc2.txt`
(ii) `cat doc2.txt doc1.txt`
(iii) `cat doc2.txt >> doc1.txt`
(iv) `cat doc1.txt > doc2.txt`
(v) `cat doc1.txt >> doc2.txt`
(vi) `cat doc2.txt > doc1.txt`

Q3 The authorization of the file txt1.txt is altered as follows. Give only the 'execute privilege' to a user who is not in user group and remove all other privileges. Select the correct set of commands to obtain this task.

(i) chmod o=x txt1.txt
(ii) chmod g+x txt1.txt
chmod g-r txt1.txt
chmod g-w txt1.txt
(iii) chmod u=x txt1.txt
(iv) chmod u+x txt1.txt

chmod u-r txt1.txt
chmod u-w txt1.txt
(v) chmod g=x txt1.txt

Q4 Select the statement which will cause the Operating System to shut-down?

(i) `gnome session quit`
(ii) `sudo init 6`
(iii) `sudo init 0`
(iv) `sudo shutdown -r now`
(v) `sudo reboot`

Q5 Consider the following code segment.

```
void ascending_swap(double *p1, double *p2);
int main(void)
{
      double values[10];
      for(int i=0;i<10;i++)
      {
            printf("\nEnter number %d: ", (i+1));
            scanf("%lf",&values[i]);
      }
      ascending_swap(values[1], values[3]); //Line 10
      return(0);
}
void ascending_swap(double *p1,double *p2) //Line 13
{
      if(*p1 > *p2) //Line 15
      {
            double temp;
            temp = *p2;
            *p2 = *p1;
            *p1 = temp;
      }
      else{}
}
```

Which of the following statement is true regarding the above code segment?

(i) Line 10 is not erroneous. *p1 in line 13, 15 tells that double value at the address pointed by pointer p1, p1 is a pointer of type double respectively.

(ii) Line 10 is erroneous. *p1 in line 13, 15 tells that p1 is a pointer of type double, double value at the address pointed by pointer p1 respectively.

(iii) Line 10 is not erroneous. *p1 in line 13, 15 tells that p1 is a pointer of type double, double value at the address pointed by pointer p1 respectively.

(iv) Line 10 is erroneous. *p1 in line 13, 15 tells that double value at the address pointed by pointer p1, p1 is a pointer of type double respectively.

Q6 Consider the following code segment.

```
union myunion
{
    char char_value;
    int int_value;
    double double_value;
};

struct mystruct
{
    char char_value;
    int int_value;
    double double_value;
};
struct mystruct initialize(char * pt1, int * pt2, double * pt3);
int main()
{
    char usr_char = "A"; int usr_int = 60; double usr_double 12.50;
    union myunion uni1;
    uni1.char_value = usr_char;
    uni1.int_value = usr_int;
    uni1.double_value = usr_double;
    struct mystruct str1 = initialize(&usr_char, &usr_int, &usr_double);
    printf("\n For Union: character is %c, integer is %d and double is %lf",
    uni1.char_value, uni1.int_value, uni1.double_value);
    printf("\n For structure: character entered is %c, integer is %d and double
    is %lf", str1.char_value, str1.int_value, str1.double_value);
    return(0);
}
struct mystruct initialize(char * pt1, int * pt2, double * pt3)
{
    struct mystruct str1;
    str1.char_value = *pt1;
    str1.int_value = *pt2;
    str1.double_value = *pt3;
    return str1;
}
```

Which of the following statement is true regarding the above code segment?

(i) For the union: The terminal screen will show **double_value** as 12.50 while other two variables will not be shown as user specified values. For the structure: The terminal screen will show **double_value** as 12.50, **int_value** as 60, **char_value** as AB

(ii) For the union: The terminal screen will show **int_value** as 60 while other two variables will not be shown as user specified values. For the structure: The terminal screen will show **double_value** as 12.50 while other two variables will not be shown as user specified values.

(iii) For the union: The terminal screen will show **char_value** as AB while other two variables will not be shown as user specified values. For the structure: The terminal screen will show **double_value** as 12.50 while other two variables will not be shown as user specified values.

(iv) For the union: The terminal screen will show **double_value** as 12.50, **int_value** as 60, **char_value** as AB. For the structure: The terminal screen will show **double_value** as 12.50, **int_value** as 60, **char_value** as AB

Q7 Consider the given pseudo code.

```
Return type thread_function1(arguments)
{
      Lock the mutex1;
      statements set 1;
      pthread_cond_signal(&cond2);
      pthread_cond_wait(&cond1,&mutex1);
      statements set 2;
      unlock the mutex1;
      exit the thread;
}
```

Which statement is false out of the following?

(i) The thread having start function as thread_function1 will have to wait until the owner thread of the mutex1 relinquish the lock before executing statements set1.

(ii) When thread_function1 is executing, some other thread sleeping based on a condition variable pointed by cond2 will wake up from sleep and lock the mutex.

(iii) Thread which calls thread_function1 will unlock the mutex temporarily until the condition variable pointed by cond2 is signaled by another thread.

(iv) If no other thread ever signals on condition variable pointed by cond1; after the thread calling thread_function1 goes to sleep; statements set 2 will never be executed.

Index number: .....

Q8 What is false about *fork( )* function?

(i) After calling this function, the new process will have a separate copy of the virtual memory address space for stack, heap and data segments.
(ii) It is not known whether the child or parent execute after a *fork( )* call.
(iii) *fork( )* function returns the process identifier of the child process inside the parent process.
(iv) It returns -1 on error and new process created will execute program text stored in different address spaces of the virtual memories of the parent and child processes.

Q9 What is true about waitpid(agr1,NULL,0) function?

(i) If arg1 is set to -1, then parent will wait until all of its children terminates. Termination occurs when after all children calls exit( ) function.
(ii) If arg1 is the process identifier of the child process, then parent will wait until that child terminates. Termination of another child will not stop the waiting of parent process.
(iii) If arg1 is the process identifier of the parent process, then child will wait until parent terminates.
(iv) This function is declared in <sys/ipc.h> file

Q10 What is true out of the following statements?

(i) The tendency of a process to refer to adjacent memory addresses of an already accessed process is known as temporal locality of reference.
(ii) The function of a page table is to map virtual memory addresses to pages in primary memory.
(iii) All of the virtual memory addresses of a particular process are stored in primary memory at a given instance.
(iv) Pointers to Machine language instructions reside in Heap segment of virtual memory of a process.

Q11 What is false out of the following statements?

(i) sleep function can suspend the execution of the process which sleep(t1) was called for t1 seconds.
(ii) When the exit( ) function is called, only the called process and all its associated threads will terminate.
(iii) When the exit( ) function is called by the parent, all its child processes and all of those child processes' associated threads will terminate.

(iv) When a process calls a function, the function's variables are stored as stack frames in virtual memory.

Q12 What is false about pipes?

(i) Pipes are stored in Kernel memory and used for communication between processes.
(ii) Function fprintf( ) called in a process can be used to send bytes to the write end of the pipe.
(iii) Function read( ) called in a process can be used to output data from the pipe to the process using read end of the pipe.
(iv) When a process reads from a pipe, the read data is not removed from the pipe and is accessible for another process to read.

Q13 What is true about the following statements?

```
int fd[2];
int x = pipe(fd);
fork( );
```

(i) **fd[0]** is associated with write end of the file.
(ii) **fd[1]** is associated with read end of the file.
(iii) If **x** is zero, a pipe is created.
(iv) Parent process and child process after calling *fork( )* will share two file descriptors **fd** with the pipe.

Q14 Select the correct order of the header files that defines each of the functions *fork( ), exit( ), pipe( ), printf( ), getpid( )* respectively?

(i) <unistd.h>, <stdlib.h>, <unistd.h>, <stdio.h>, <unistd.h>
(ii) <stdlib.h>, <unistd.h>, <stdio.h>, <unistd.h>, <errno.h>
(iii) <unistd.h>, <sys/wait.h>, <unistd.h>, <stdio.h>, <errno.h>
(iv) <unistd.h>, <stdio.h>, <unistd.h>, <stdlib.h>, <unistd.h>

Q15 What is true about int msgget(key_t key1, int msgflag)

(i) msgget( ) resembles the function write( ) in File I/O.
(ii) If IPC_EXCL is passed to msgflag and a message queue created using key1 already exists, msgget( ) will return -1.
(iii) passing 0666 to msgflag will cause the user group to have read only privileges.
(iv) Passing the phrase IPC_CREAT will cause the msgget( )to return an error if a message queue produced using the given key exists and create a message queue if it does not exist.

Index number: .....

(Q16) A program written in C contains the following statements.

```
int fp1=open("doc1.txt",O_RDWR|O_CREAT|O_TRUNC,S_IRUSR|S_IWUSR|S_IRGRP|
S_IWGRP|S_IROTH|S_IWOTH);
printf("%d", fp1);
```

Select the false statement regarding the given statement.

(i) If doc1.txt does not exist; value of fp1 will be -1
(ii) If doc1.txt exists; the content will be deleted
(iii) All user, user group and other users can read from and write to doc1.txt
(iv) If doc1.txt does not exist; a new file will be created and the value of fp1 will be positive.
(v) If doc1.txt cannot be opened; value of fp1 will be -1

Q17 A text file **txt1.txt** contains the following text.

"Hello this is operating system programming end semester examination"

Following set of instructions are run inside a program.

```
int fd1;
fd1 = open("txt1.txt", O_RDONLY,0666);
char buf1[10];
lseek(fd1,14,SEEK_SET);
lseek(fd1,10,SEEK_CUR);
read(fd1,buf1,10);
```

The content of the buffer buf1 is given by;

(i) "operating "
(ii) "system pro"
(iii) "Hello this "
(iv) "programmin"

Q18 What is true out of the following?

(i) sem_wait( ) and sem_post( ) called by a thread function has equivalent behavior compared to pthread_mutex_lock( ) and pthread_mutex_unlock( ) respectively.
(ii) Semaphores are stored in user space so that they can be used for process synchronization.
(iii) Mutually exclusive locks are always owned by a thread.
(iv) A Mutex variable initialized using PTHREAD_MUTEX_INITIALIZER can be used for process synchronization.

Index number: .....

Q19 What is false out of the following?

(i) Multiple semaphores initialed to zero can be used for thread serialization.
(ii) Only the owner thread of a mutex can unlock a mutex
(iii) Only the process or thread which called sem_wait( ) can call sem_post( ).
(iv) Using mutually exclusive locks in Simultaneous Multi-Threading (SMT) without initializing may lead to deadlocks.

Q20 What is false out of the following?

(i) Thread synchronization is not mandatory when each thread update its own local variable.
(ii) A concurrency control mechanism is required for hyper-threading applications that update the data segment variable of virtual address space.
(iii) Multiple threads updating a static variable defined inside a function do not cause inconsistent updates.
(iv) A semaphore can be used in shared memory to control simultaneous updates of the shared memory.