Started on	Tuesday, 6 May 2025, 3:18 PM
State	Finished
Completed on	Tuesday, 6 May 2025, 3:56 PM
Time taken	38 mins 8 secs
Marks	31.83/50.00
Grade	**7.96** out of 12.50 (**63.67**%)

Question **1**

Correct

Mark 1.00 out of 1.00

The difference between trap and error is:

Select one or more:

- a. the error is issued by hardware and the trap by software
- b. there is no significant difference
- c. errors are generally reported asynchronously and traps synchronously ✔
- d. after an error, there is usually no return to the program, and after a trap, yes ✔

Twoja odpowiedź jest poprawna.

The correct answers are: after an error, there is usually no return to the program, and after a trap, yes, errors are generally reported asynchronously and traps synchronously

Question **2**

Correct

Mark 1.00 out of 1.00

In the case of hardware interrupt summation ("wire or"):

Select one or more:

- a. the processor can programatically poll devices about issuing an interrupt ✔
- b. the interrupt circuit can query devices about issuing an interrupt
- c. the bus driver can poll devices about issuing an interrupt ✔
- d. the interrupt source is identified by a special bus signal

Twoja odpowiedź jest poprawna.

The correct answers are: the processor can programatically poll devices about issuing an interrupt, the bus driver can poll devices about issuing an interrupt

Question 3

Incorrect

Mark 0.00 out of 1.00

Precise interrupts are:

Select one or more:

- a. accepted only in a stable state between the execution of successive instructions ✔
- b. unblocked only in a stable state between the execution of successive instructions ✖
- c. only issued in a stable state between executions of subsequent instructions ✖
- d. transferred only in a stable state between the execution of successive instructions instrukcji ✖

Twoja odpowiedź jest niepoprawna.

The correct answer is: accepted only in a stable state between the execution of successive instructions

Question 4

Incorrect

Mark 0.00 out of 1.00

If the loader inserts a program with absolute addresses into a fixed address space, it is called:

- a. Dynamic relocation
- b. Static relocation ✖
- c. Dynamic loading
- d. Static loading

The correct answer is: Dynamic relocation

Question 5

Correct

Mark 1.00 out of 1.00

Dynamic relocation requires the use of:

Select one or more:

- a. base register (DATUM) ✔
- b. limit register
- c. status register
- d. program counter

Twoja odpowiedź jest poprawna.

The correct answer is: base register (DATUM)

Question 6

Correct

Mark 1.00 out of 1.00

If the compiler prepares a program with absolute addresses to be loaded in a fixed address space, it is called:

- a. Dynamic compiling
- b. Static relocation ✔
- c. Static compiling
- d. Dynamic relocation

The correct answer is: Static relocation

Question 7

Partially correct

Mark 0.50 out of 1.00

Dynamic relocation is performed by:

- a. Paging system
- b. Linker
- c. Special registers (DATUM) ✔
- d. Loader

The correct answers are: Special registers (DATUM), Paging system

Question 8

Correct

Mark 1.00 out of 1.00

The scheduling goal, which is to occupy processors as efficiently as possible, is:

Select one or more:

- a. response time
- b. utilization ✔
- c. productivity

Twoja odpowiedź jest poprawna.

The correct answer is: utilization

Question 9

Incorrect

Mark 0.00 out of 1.00

Which of the following memory allocation schemes can cause external fragmentation?

- a. Multiple contiguous fixed partitions of various sizes ✖
- b. Segmentation ✔
- c. Paging
- d. Multiple contiguous fixed partitions of equal size
- e. Sweeping ✔

The correct answers are: Segmentation, Sweeping

Question 10

Partially correct

Mark 0.33 out of 1.00

Onion algorithm:

Select one or more:

- a. Causes external fragmentation
- b. Reduces external fragmentation by aligning the allocation order with the process hierarchy ✔
- c. Causes internal fragmentation

Twoja odpowiedź jest częściowo poprawna.

You have correctly selected 1.

The correct answers are: Causes internal fragmentation, Causes external fragmentation, Reduces external fragmentation by aligning the allocation order with the process hierarchy

Question 11

Partially correct

Mark 0.50 out of 1.00

Which of the following memory allocation schemes cause external fragmentation?

- a. Sweeping
- b. Paging
- c. Segmentation ✔
- d. Multiple contiguous fixed partitions of equal size

The correct answers are: Segmentation, Sweeping

Question 12

Correct

Mark 1.00 out of 1.00

Context switch is:

Select one or more:

- a. switching to the system stack
- b. writing registers to the task stack and retrieving them from another task stack ✔
- c. extracode execution
- d. calling the kernel of the operating system

Twoja odpowiedź jest poprawna.

The correct answer is: writing registers to the task stack and retrieving them from another task stack

Question 13

Correct

Mark 1.00 out of 1.00

The system stack must provide space for:

Select one or more:

- a. CPU registers
- b. Processor registers in as many copies as there are interrupt lines and possible software interrupts
- c. Processor registers in as many copies as there are interrupt lines +1 ✔
- d. Processor registers in as many copies as there are devices in the system +1

Twoja odpowiedź jest poprawna.

The correct answer is: Processor registers in as many copies as there are interrupt lines +1

Question 14

Incorrect

Mark 0.00 out of 1.00

What is a scheduler?

Select one or more:

- a. procedure that schedules frame release in the page replace algorithm
- b. memory allocation procedure
- c. a kernel routine that selects a task to execute ✔
- d. the system process that allocates the processor ✖

Twoja odpowiedź jest niepoprawna.

The correct answer is: a kernel routine that selects a task to execute

Question 15

Correct

Mark 1.00 out of 1.00

Which of the following applies to user-level threads?

- a. The organization of user-level threads is specific to the operating system.
- b. User-level threads cost no execution time in system mode. ✔
- c. User-level threads require their descriptors in the kernel.
- d. User-level threads can themselves be multi-threaded.

The correct answer is: User-level threads cost no execution time in system mode.

Question 16

Correct

Mark 1.00 out of 1.00

The thread is also called:

- a. lightweight process ✔
- b. data process
- c. overlay process
- d. heavy process

The correct answer is: lightweight process

Question 17

Partially correct

Mark 0.50 out of 1.00

System/user threads:

Select one or more:

- [ ] a. User-level threads share the same stack.
- [ ] b. User-level threads share the same execution context.
- [x] c. User-level thread descriptors are stored in the address space of the program. ✔

Twoja odpowiedź jest częściowo poprawna.

You have correctly selected 1.
The correct answers are: User-level thread descriptors are stored in the address space of the program., User-level threads share the same execution context.

Question 18

Incorrect

Mark 0.00 out of 1.00

What is the average time in the system for tasks in the batch incoming in this order, using FCFS algorithm?

The system is equipped with 3 processors

task	1	2	3	4
processing time	3.6	6	2.6	1.3

Answer: 3.7 ✖

The correct answer is: 4.0

Question 19

Correct

Mark 1.00 out of 1.00

We have the tasks with the following priorities in the access to a resource (a larger number is a higher priority):

Process number	1	2	3	4	5	6
Priority	37	38	29	30	32	25

The scheduling policy is preemptive, with the decision time constraint based on time slicing.

The current process is number 4. What process will get the resource in the nearest decision? Provide its priority.

Answer: 38 ✔

The correct answer is: 38

Question 20

Incorrect

Mark 0.00 out of 1.00

What is the average time in the system for tasks in the batch incoming in this order, using FCFS algorithm?

The system is equipped with 4 processors

task	1	2	3	4
processing time	3.2	6.2	2.7	1.6

Answer: 3.8 ✖

The correct answer is: 3.4

Question 21

Correct

Mark 5.00 out of 5.00

page	M	R	belongs to process
1	1	0	4
2	0	0	3
3	1	1	3
4	0	1	2
5	1	0	6
6	1	0	6
7	0	0	5
8	1	1	4

Using the above table of bits M and R for the pages in NRU swapping, with the priority frame allocation rule, which page will be sent to the disk first? The pages are scanned starting from the top. A process number is its priority (the smaller number, the higher priority). The process for which the frame is needed is 4. Provide a page number.

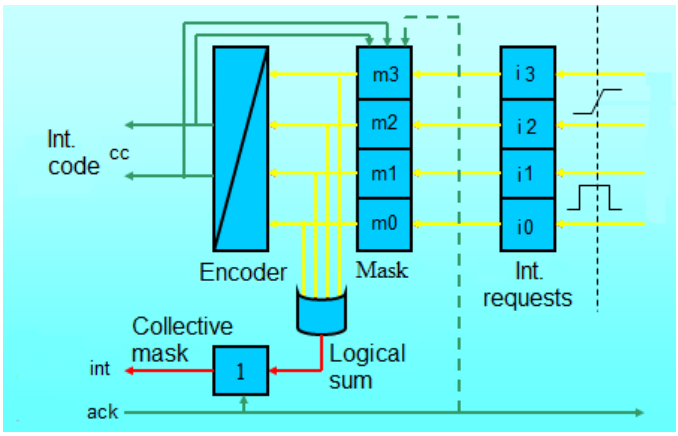
Answer: 7 ✔

The correct answer is: 7

Question 22

Incorrect

Mark 0.00 out of 5.00



In the given interrupt controller structure, the interrupt number cc reported to the processor is equal to 0. What will be the form of the interrupt mask (from m3 to m0) after the processor confirms the reception of this interrupt?

The interrupt with index 3 has the highest priority.

Enter the mask in the form .m3m2m1m0 (preceded by a point), for example .0101

If this is not possible, enter -.0001.

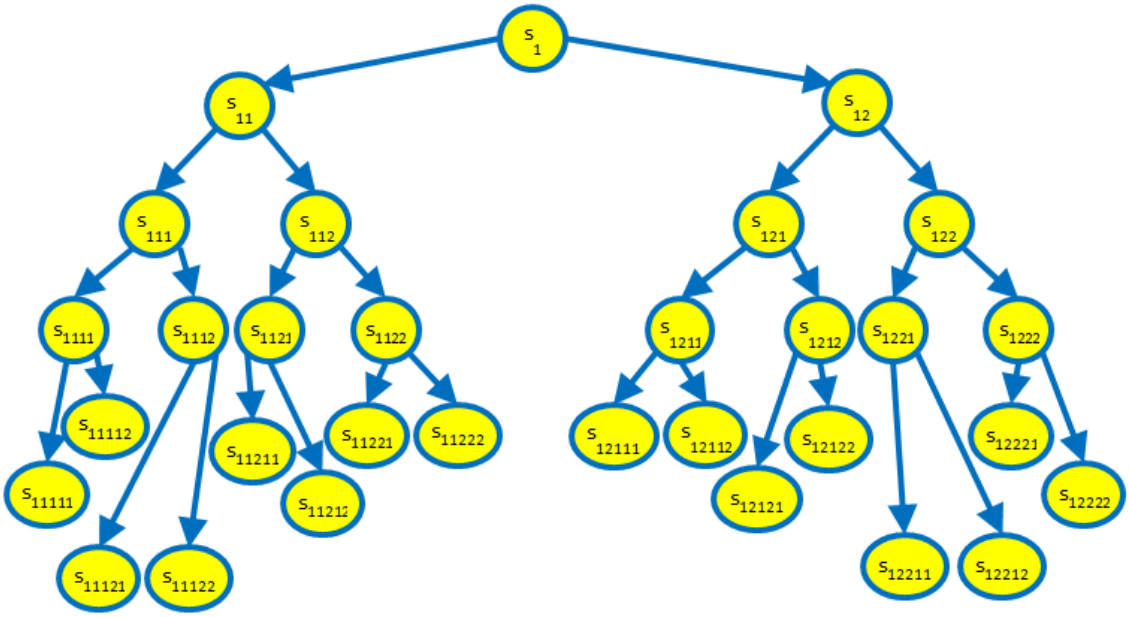
Answer: .1111 ✖

The correct answer is: 0.1110

Question 23

Correct

Mark 5.00 out of 5.00



Above is the nesting structure of subroutines in some program. Subroutine s12211 calls its "parent" in the hierarchy. To what stack frame (which subroutine) will the dynamic link be made in the called subroutine frame?

Enter the name of the subroutine with the index, without the letter "s", e.g. for s11122 enter 11122

Answer: 12211 ✔

The correct answer is: 12211

Question **24**

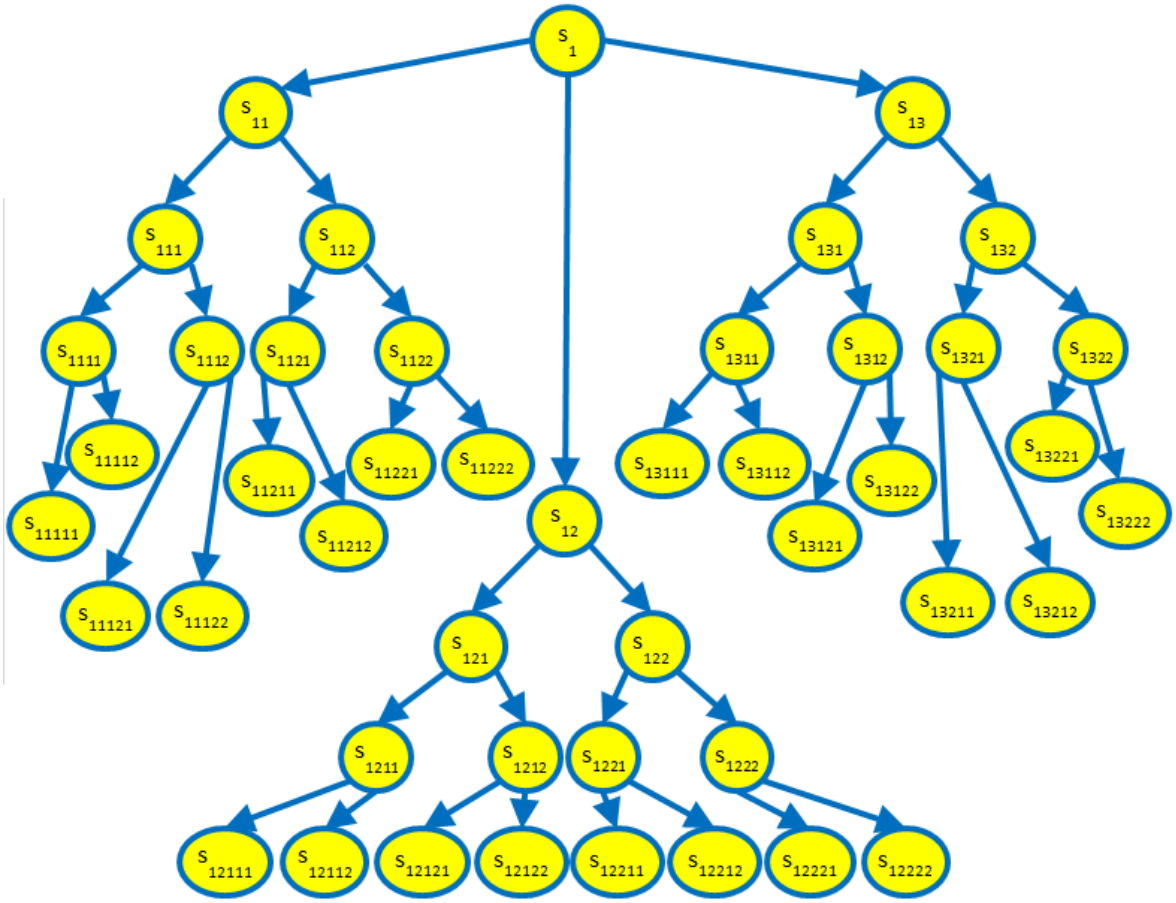
Incorrect

Mark 0.00 out of 5.00

Below is the nesting structure of subroutines in some program.

The stack frame of which subroutine (specify the index) will be pointed by the static link in the frame of the subroutine S1211, that is called from the subroutine S12112?

If the call is not possible, enter index -1.



Answer: -1 ✖

The correct answer is: 121

Question 25

Correct

Mark 5.00 out of 5.00

What is the average time in the system for tasks in the batch, using SJF algorithm?

task	1	2	3	4
processing time	3.4	6.4	2.7	1.1

Answer: 6.4 ✔

The correct answer is: 6.4

Question 26

Correct

Mark 5.00 out of 5.00

The organization of memory and processor is word-based. A word means 16 bits (int also takes 1 16-bit word). Hexadecimal values are preceded by a # character.

variable a has the value #a0c0

top of the stack (full descending, i.e. the stack pointer points to the most recently put element on the stack, and the stack expands towards lower addresses) #c100

stack frame before subroutine B call #c1010

address of subroutine B #1021

Calling rules: Parameters are put on the stack in accordance with the convention of the C language, i.e. starting from the last one, without a static connection, the result of the function is passed in registers. The stack is shown after subroutine B is called by itself (recursively), at label point C. Subroutine B is called for the first time B(a-2); somewhere in the program from address #1011. Label C has address #1050.

regardless of the programming language, subroutine B has the form

```
B(int i);
{
        int p=i+1;
C:      B(i-3);
}
```

The content of the cell at address #c0ff: parameter ✔

Address	content	
#c100	????	
#c0ff	#a0be	
#c0fe	#1012	
#c0fd	#c010	
#c0fc	#a0bf	
#c0fb	#a0bb	
#c0fa	#1051	
#c0f9	#c0fd	
#c0f8	#a0bc	
#c0f7	????	
#c0f6	????	

Twoja odpowiedź jest poprawna.

The correct answer is:

The organization of memory and processor is word-based. A word means 16 bits (int also takes 1 16-bit word). Hexadecimal values are preceded by a # character.

variable a has the value #a0c0

top of the stack (full descending, i.e. the stack pointer points to the most recently put element on the stack, and the stack expands towards lower addresses) #c100

stack frame before subroutine B call #c1010

address of subroutine B #1021

Calling rules: Parameters are put on the stack in accordance with the convention of the C language, i.e. starting from the last one, without a static connection, the result of the function is passed in registers. The stack is shown after subroutine B is called by itself (recursively), at label point C. Subroutine B is called for the first time B(a-2); somewhere in the program from address #1011. Label C has address #1050.

regardless of the programming language, subroutine B has the form

```
B(int i);
{
        int p=i+1;
C:      B(i-3);
}
```

The content of the cell at address #c0ff:[parameter]

Address	content	
#c100	????	
#c0ff	#a0be	
#c0fe	#1012	
#c0fd	#c010	
#c0fc	#a0bf	
#c0fb	#a0bb	
#c0fa	#1051	
#c0f9	#c0fd	
#c0f8	#a0bc	
#c0f7	????	
#c0f6	????	

Started on	Tuesday, 10 June 2025, 12:28 AM
State	Finished
Completed on	Tuesday, 10 June 2025, 12:32 AM
Time taken	4 mins 19 secs
Marks	2.00/4.00
Grade	**0.01** out of 0.01 (**50**%)

Question **1**

Incorrect

Mark 0.00 out of 1.00

Assuming that memory cells are 8-byte, the page number in the address field is 13 bits, the offset is 12 bits, the frame number is 11 bits, and all entries in the TIS page index table are on an 8-byte word boundary, please specify the maximum size of the physical memory in MB

Answer: 4 ✖

The number of frames is 2048 (11 bits), and the frame size is 4096 cells (12-bit address), so the maximum number of cells in physical memory is 8M cells (1023).
Because the cells are 8-byte, the maximum memory size is 8Mx8B=64MB

The correct answer is: 64

Question **2**

Incorrect

Mark 0.00 out of 1.00

Assuming that memory cells are 4-byte, the page number in the address field is 13 bits, the offset is 12 bits, the frame number is 11 bits, and all entries in the TIS page index table are on an 8-byte word boundary, please specify the maximum size of the physical memory in MB

Answer: 8 ✖

The number of pages is 8096 (13 bits), and the frame size is 4096 cells (12-bit address), so the maximum number of cells in virtual memory is 32M cells (2^{25}).
Because the cells are 4-byte, the maximum memory size is 32Mx4B=128MB

The correct answer is: 128

Question 3

Correct

Mark 1.00 out of 1.00

The virtual address consists of 8b page number and 8b offset. The page index table is shown below (index, content). For decimal address 898, binary 0000 0011 1000 0011, enter the physical address in the form: frame number.offset (as decimal numbers, offset in 3 digits). For example, for a physical address consisting of frame 0 and offset 18, specify 0.018. If there is no physical address for the given virtual address, then -1 should be specified.

3	15
2	0
1	-1
0	-1

Answer: 15.131 ✔

The address is divided into 8b page number: 0000 0011, and 8b offset: 1000 0011.

In decimal, they are 3 and 131.

Under index 3, in PIT there is frame number 15. Therefore, the solution is 15.131

The correct answer is: 15.131

Question 4

Correct

Mark 1.00 out of 1.00

Assuming that memory cells are 8-byte, the page number in the address field is 13 bits, the offset is 12 bits, the frame number is 11 bits, and all entries in the TIS page index table are on a 4-byte word boundary, please specify the size of the Page Index Table PIT in KB

Answer: 16 ✔

The number of pages is 4096 (12 bits), and the PIT cell size is 4 bytes, so the size of PIT is 4Kx4B=16KB

The correct answer is: 16

Started on	Tuesday, 10 June 2025, 12:13 AM
State	Finished
Completed on	Tuesday, 10 June 2025, 12:22 AM
Time taken	9 mins 4 secs
Marks	3.00/5.00
Grade	**0.01** out of 0.01 (**60**%)

Question **1**

Incorrect

Mark 0.00 out of 1.00

For the disk operation scheduling Cyclic SCAN method, the currently being executed operation is in 37 cylinder. The direction in the Cyclic SCAN method is descending. The next scheduled operations (in the order of their queuing) are:

Operation number	1	2	3
Cylinder number	48	29	34

Enter the number of the next operation.

Answer: 2 ✖

The next operation is in a closest cylinder in the descending direction, that is number 3 - cylinder 27

The correct answer is: 3

Question 2

Correct

Mark 1.00 out of 1.00

For the disk operation scheduling Cyclic SCAN method, the currently being executed operation is in 21 cylinder. The direction in the Cyclic SCAN method is ascending. The next scheduled operations (in the order of their queuing) are:

Operation number	1	2	3
Cylinder number	18	29	34

Which disk operation will directly precede the operation on cylinder 18? Provide the number of the operation.

Answer: 3 ✔

After executing all operations in cylinders of greater numbers than current (or equal), the head will be moved to the cylinder of the smallest number in the orders (which is 18). The greatest cylinder number in the orders is 34 - order number 3

The correct answer is: 3

Question 3

Incorrect

Mark 0.00 out of 1.00

For the disk operation scheduling SCAN method, the currently being executed operation is in 30 cylinder. The direction in the SCAN method is descending. The next scheduled operations (in the order of their queuing) are:

Operation number	1	2	3
Cylinder number	48	21	34

Provide the number of the next operation.

Answer: 3 ✖

Because the current direction is descending, the operation will be chosen in the closest cylinder with a smaller (or equal) number: cylinder 21 (number 2).

The correct answer is: 2

Question **4**

Correct

Mark 1.00 out of 1.00

For the disk operation scheduling SCAN method, the currently being executed operation is in 31 cylinder. The direction in the SCAN method is descending. The next scheduled operations (in the order of their queuing) are:

Operation number	1	2	3
Cylinder number	48	29	34

After which disk operation the current scan direction will change? Provide the number of the operation after which the direction will change.

Answer: 2 ✔

All the operations on smaller cylinder numbers will be executed before the direction change. The smallest one is in cylinder 29 - number 2

The correct answer is: 2

Question **5**

Correct

Mark 1.00 out of 1.00

For the disk operation scheduling SSTF method, the currently being executed operation is in 30 cylinder. The next scheduled operations (in the order of their queuing) are:

Operation number	1	2	3
Cylinder number	48	29	34

Provide the number of the next operation.

Answer: 2 ✔

In SSTF, the next operation will be executed in the cylinder closest to the current one: this is cylinder 29 (operation 2).

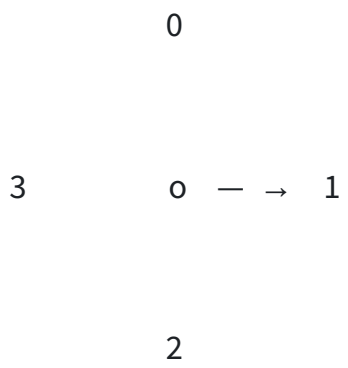
The correct answer is: 2

Started on	Tuesday, 10 June 2025, 12:04 AM
State	Finished
Completed on	Tuesday, 10 June 2025, 12:12 AM
Time taken	8 mins 23 secs
Marks	5.00/6.00
Grade	**8.33** out of 10.00 (**83.33**%)

Question **1**

Correct

Mark 1.00 out of 1.00



page	0	1	2	3
R bit	1	1	0	1

Using the above list for the Clock swapping algorithm, which page will be sent to the disk first? Provide a page number.

The current position of the "arrow" is 1, and the algorithm works clockwise.

Answer: 2 ✔

The algorithm starts from the current position - page 1 . It has R=1, so it receives R-9 and the clock hand moves to page 2. It has R=0, so this page is selected for swapping.

The correct answer is: 2

Question 2

Incorrect

Mark 0.00 out of 1.00

page nr	1	2	3	4
loaded	6	12	33	56
R bit	1	1	1	0

Using the above list for the FIFO-second chance swapping algorithm, what will be the bottom row after page replacement execution? The middle row is the number of a time slice in which a page was loaded to a frame. Provide a sequence of bits R for the pages; for example, before page replacement, the bottom row has the image 1110. We assume that the loaded page gets R=0.

Answer: 0 ✖

The 2-chance algorithm first looks at R in the oldest page (1). It is 1, so this page is moved to the opposite end of the queue, and R is cleared to this page. At this moment the bottom row is 11100. The algorithm continues searching, and finally, it finds page 4 with R=0 and replaces it. The loaded page receives R=0, so the bottom row remains 1100.

The correct answer is: 1100

Question 3

Correct

Mark 1.00 out of 1.00

page	history of R
0	11
1	01
2	00
3	10

Using the above table of the history of R bit for the pages in LFU swapping, which page will be sent to the disk first? The oldest bit R is on the left. Provide a page number.

Answer: 2 ✔

Page 2 has the smallest number of 1s, so it will be swapped.

The correct answer is: 2

Question **4**

Correct

Mark 1.00 out of 1.00

		page		
		0	1	2
page	0	0	1	0
	1	0	0	1
	2	1	1	1

Using the above table for hardware-based implementation of LRU swapping, which page will be sent to the disk first? Provide a page number.

Answer: 1 ✔

Row 1, interpreted as a binary number (001), is the smallest one, so page 1 will be swapped first.

The correct answer is: 1

Question **5**

Correct

Mark 1.00 out of 1.00

page	history of R
0	11
1	00
2	01
3	10

Using the above table of the history of R bit for the pages in LRU swapping, which page will be sent to the disk first? The oldest bit R is on the left. Provide a page number.

Answer: 1 ✔

Page 1 has the longest number of 0s, so it will be swapped.

The correct answer is: 1

Question 6

Correct

Mark 1.00 out of 1.00

page	Last used	Bit R
1	89	1
2	91	0
3	87	0
4	92	0

Using the above table of the history of R bit for the pages in Workset swapping algorithm, which page will be sent to the disk first? The current time slice number is 95, and the time range τ for the workset is 5 (the threshold 95-τ=90). Page scan starts from the top. Provide a page number to be sent to the disk.

Answer: 3 ✔

The first page with R=0 out of the workset is 3 with timestamp 87.

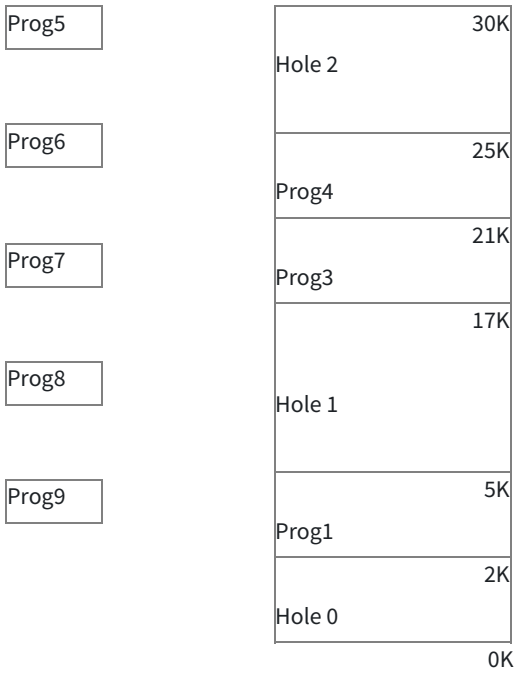
The correct answer is: 3

Started on	Tuesday, 6 May 2025, 11:28 AM
State	Finished
Completed on	Tuesday, 6 May 2025, 11:33 AM
Time taken	5 mins 7 secs
Marks	3.00/7.00
Grade	**0.00** out of 0.01 (**42.86**%)

Question **1**

Correct

Mark 1.00 out of 1.00



The numbers on the right are the addresses that finish the program/hole areas, so the area size is the difference between the upper address and the lower address.

In the above memory allocation state, 3 programs are already in memory, and 5 programs are waiting to be loaded into the memory, in the order of their numbers,. The programs waiting to be loaded into memory have the following sizes:

prog5 - 2K
prog6 - 6K
prog7 - 4K
prog8 - 7K
prog9 - 4K

The memory is allocated to the programs in the first-fit rule, starting from lower addresses, with making a new hole if the allocated block is larger than the demand.

The programs are loaded into the memory in the order of their numbers (from 5 to 9).

Specify, which program will cause the memory compaction. Provide only the program number. If the compaction will not be needed, provide 0.

Answer: 8 ✔

Program 5 fits to hole 0, leaving no room.

Program 6 fits to hole 1, reducing this hole to size 6K.

Program 7 fits to hole 1, reducing its size to 2K.

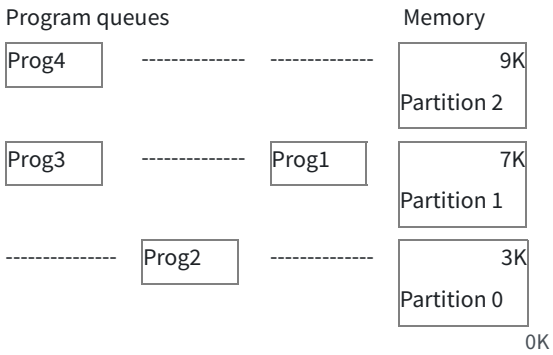
Program 8 does not fit to hole 1 (2K) and to hole 2 (5K), causing the compaction.

The correct answer is: 8

Question **2**

Incorrect

Mark 0.00 out of 1.00



The numbers on the right are the addresses that finish the partitions, so the partition size is the difference between the upper address and the lower address.

In the above memory division, the programs waiting to be loaded into memory have the following sizes:

prog1 - 4K
prog2 - 3K
prog3 - 2K
prog4 - 4K

The programs are loaded into memory partitions (if possible) in the order of their numbers, from 1 to 4.

Specify, which program will face the internal fragmentation problem as the first. Provide only the program number. If the internal fragmentation cannot occur, enter 0.

Answer: 0 ✖

Program 1 fits the partition 1 size - no internal fragmentation.

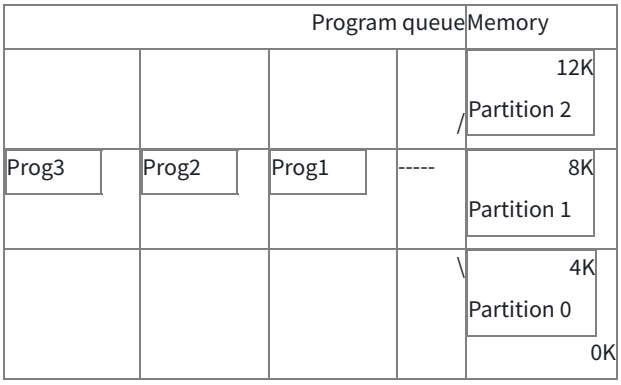
Program 2 is smaller than partition 0 size - it causes internal fragmentation.

The correct answer is: 2

Question 3

Correct

Mark 1.00 out of 1.00



The numbers on the right are the addresses that finish the partitions, so the partition size is the difference between the upper address and the lower address.

In the above memory division, the programs waiting to be loaded into memory have the following sizes:

prog1 - 5K
prog2 - 4K
prog3 - 3K

Programs are loaded into memory partitions (if possible) in the order of their numbers, from 1 to 3.

Specify, which program will face the internal fragmentation problem as the first. Provide only the program number. If the internal fragmentation cannot occur, enter 0.

Answer: 3 ✔

Program 1 is greater than the partition size - it will not be executed

Program 2 fits the partition size - no internal fragmentation.

Program 3 is smaller than the partition size - it causes internal fragmentation.

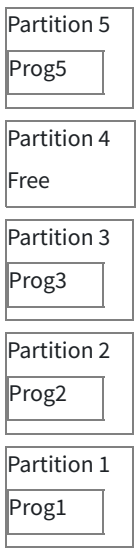
The correct answer is: 3

Question **4**

Incorrect

Mark 0.00 out of 1.00

Memory



In the above memory allocation state, partition 4 is a hole, and there are partitions 1...3,5 that are occupied by programs. Starting from the current while, some programs finish their work and release their partitions in the order: prog5, prog3, prog1. Free neighboring partitions are merged.

Specify the first program termination that causes external fragmentation. Provide the number of this program. If external fragmentation will not occur, enter 0.

Answer: 3 ✖

Program 5 and Program 3 stick to the free partition, so their partitions are merged with Partition 4.

Partition 1 is distant from the hole (there is Partition 2 in between), so the termination of Program 1 causes a separate hole.

Occurring of two separate memory holes causes eternal fragmentation.

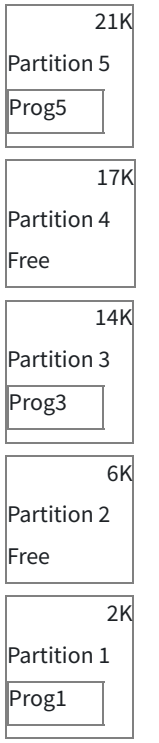
The correct answer is: 1

Question **5**

Incorrect

Mark 0.00 out of 1.00

Memory



The numbers on the right are the addresses that finish the partitions, so the partition size is the difference between the upper address and the lower address.

In the above memory allocation state, 4 programs are already in memory, and 5th program is waiting to be loaded into the memory.

The memory is allocated to the programs in the best-fit rule, without making a new hole if the allocated block is larger than the demand.

What will be the internal fragmentation after loading the program Prog4 of size 2K into memory?

Answer: 2 ✖

Both Partition 2 and Partition 4 fit the requirements of Prog4, but Partition 4 has a smaller size, so it will receive the program, and the fragmentation will be 1K (3-2).

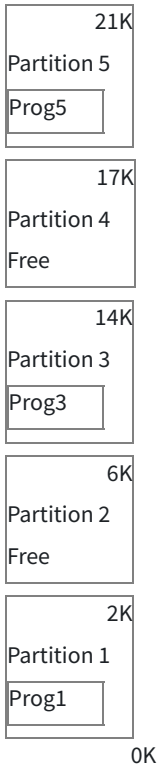
The correct answer is: 1

Question 6

Correct

Mark 1.00 out of 1.00

Memory



The numbers on the right are the addresses that finish the partitions, so the partition size is the difference between the upper address and the lower address.

In the above memory allocation state, 4 programs are already in memory, and 5th program is waiting to be loaded into the memory.

The memory is allocated to the programs in the first-fit rule, without making a new hole if the allocated block is larger than the demand. The memory is scanned for the fitting hole starting from the higher addresses.

What will be the internal fragmentation after loading the program Prog4 of size 4K into memory?

Answer: 0 ✔

Partition 4 has he size 3K, so it is too small. Partition 2 fits the requirement, so the program arrives here. The partition size is equal than the program, so the fragmentation will be 0 (4-4).

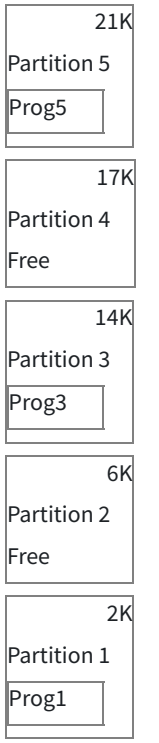
The correct answer is: 0

Question 7

Incorrect

Mark 0.00 out of 1.00

Memory



The numbers on the right are the addresses that finish the partitions, so the partition size is the difference between the upper address and the lower address.

In the above memory allocation state, 4 programs are already in memory, and 5th program is waiting to be loaded into the memory.

The memory is allocated to the programs in the worst-fit rule, without making a new hole if the allocated block is larger than the demand.

What will be the internal fragmentation after loading the program Prog4 of size 2K into memory?

Answer: 6 ✖

Both Partition 2 and Partition 4 fit the requirements of Prog4, but Partition 4 has a larger size, so it will receive the program, and the fragmentation will be 2K (4-2).

The correct answer is: 2

Started on	Tuesday, 6 May 2025, 10:58 AM
State	Finished
Completed on	Tuesday, 6 May 2025, 11:23 AM
Time taken	24 mins 59 secs
Marks	8.48/10.00
Grade	**0.01** out of 0.01 (**84.78**%)

Question **1**

Partially correct

Mark 0.54 out of 1.00

We have a buffer of capacity N=70, and two processes of producers A and B. Producer A produces the letter A, and producer B produces the letter B.

Complete the code of the prodA and prodB functions, which will be cyclically called by producer A and producer B, respectively.

Each function call concerns 1 element. Semaphores with initialization, p, v operations should be used.

When using semaphores, it is necessary to ensure synchronization of access to the buffer so that letters are placed in it according to the sequence:

AABBBAABBBAABBBAABBB .....

binsem mutex = 1, semA = 0; semB = 0;

int count= 0 ✔ ; countA= 0 ✔ ; countB= 0 ✖ ;

bool waitA=false, waitB=false;

```
void prodA() {
  mutex.P;
  if ( (count == N) || ( countA == 2 ✔ )) {
    waitA = true;
    mutex.V;
    semA.P;
    waitA = false;
  }
  //produce an element
  count += 1;
  countA += 1 ✔ ;
  if (waitB && (count<N) && ( countB == 3 ✖ )) {
    countA ✖ = 3 ✖ ;
    semB.V;
    mutex.P;
  }
  mutex.V;
}

void prodB() {
```

```
  mutex.P;
  if ( (count == N) || ( countB == 3 ✔ )) {
    waitB = true;
    mutex.V;
    semB.P;
    waitB = false;
  }
  //produce an element
  count += 1;
  countB += 1 ✔ ;
  if (waitA && (count<N) && ( countB == 0 ✖ )) {
    countA ✔ = -1 ✖ ;
    semA.V;
    mutex.P;
  }
  mutex.V
}
```

Twoja odpowiedź jest częściowo poprawna.

You have correctly selected 7.

The correct answer is:

We have a buffer of capacity N=70, and two processes of producers A and B. Producer A produces the letter A, and producer B produces the letter B.

Complete the code of the prodA and prodB functions, which will be cyclically called by producer A and producer B, respectively.

Each function call concerns 1 element. Semaphores with initialization, p, v operations should be used.

When using semaphores, it is necessary to ensure synchronization of access to the buffer so that letters are placed in it according to the sequence:

AABBBAABBBAABBBAABBB .....

binsem mutex = 1, semA = 0; semB = 0;

int count=[0]; countA=[0]; countB=[3];

bool waitA=false, waitB=false;

```
void prodA() {
  mutex.P;
  if ( (count == N) || ([countA == 2])) {
    waitA = true;
    mutex.V;
    semA.P;
```

```
        waitA = false;
    }
    //produce an element
    count += 1;
    [countA += 1];
    if (waitB && (count<N) && ([countA == 2])) {
        [countB] = [0];
        semB.V;
        mutex.P;
    }
    mutex.V;
}

void prodB() {
    mutex.P;
    if ( (count == N) || ([countB == 3])) {
        waitB = true;
        mutex.V;
        semB.P;
        waitB = false;
    }
    //produce an element
    count += 1;
    [countB += 1];
    if (waitA && (count<N) && ([countB == 3])) {
        [countA] = [0];
        semA.V;
        mutex.P;
    }
    mutex.V
}
```

Question 2

Correct

Mark 1.00 out of 1.00

We have a buffer of capacity N=25, and two processes of producers A and B. Producer A produces the letter A, and producer B produces the letter B.

Complete the code of the prodA and prodB functions, which will be cyclically called by producer A and producer B, respectively.

Each function call concerns 1 element. Semaphores with init, p, v operations should be used.

When using semaphores, it is necessary to ensure synchronization of access to the buffer so that letters are placed in it according to the sequence:

AABBBBAABBBBAABBBBAABBBB .....

binsem semA = 1 ✔ , semB = 0 ✔ ;

sem semFull = 25 ✔ , semEmp = 0 ✔ ;

int countA = 0 ✔ , countB = 0 ✔ ;

```
void prodA() {
  semFull.P;
  semA.P;
  //produce an element
  countA += 1 ✔ ;
  if (countA == 2 ✔ )
  {  countB = 0 ✔ ; semB.V;   }
   else
     semA.V;
  semEmp.V;
}

void prodB() {
  semFull.P;
  semB.P;
  //produce an element
  countB += 1 ✔ ;
  if (countB == 4 ✔ )
  {  countA = 0 ✔ ; semA.V;   }
```

```
  else
    semB.V;
  semEmp.V;
}
```

Twoja odpowiedź jest poprawna.

The correct answer is:
We have a buffer of capacity N=25, and two processes of producers A and B. Producer A produces the letter A, and producer B produces the letter B.

Complete the code of the prodA and prodB functions, which will be cyclically called by producer A and producer B, respectively.

Each function call concerns 1 element. Semaphores with init, p, v operations should be used.

When using semaphores, it is necessary to ensure synchronization of access to the buffer so that letters are placed in it according to the sequence:

AABBBBAABBBBAABBBBAABBBB .....

```
binsem semA = [1], semB = [0];
sem semFull = [25], semEmp = [0];
int countA = [0], countB = [0];

void prodA() {
  semFull.P;
  semA.P;
  //produce an element
  countA += [1];
  if (countA == [2])
  {  countB = [0]; semB.V;  }
   else
    semA.V;
  semEmp.V;
}

void prodB() {
  semFull.P;
  semB.P;
  //produce an element
  countB += [1];
  if (countB == [4])
  {  countA = [0]; semA.V;  }
  else
    semB.V;
  semEmp.V;
}
```

Question 3

Partially correct

Mark 0.73 out of 1.00

We have a buffer of capacity N=25, and two processes of producers A and B. Producer A produces the letter A, and producer B produces the letter B.

Complete the code of the prodA and prodB functions, which will be cyclically called by producer A and producer B, respectively.

Each function call concerns 1 element. Semaphores with init, p, v operations should be used.

When using semaphores, it is necessary to ensure synchronization of access to the buffer so that letters are placed in it according to the sequence:

AABBBBAABBBBAABBBBAABBBB .....

binsem semA = 1 ✔ , semB = 0 ✔ ;

sem semFull = 25 ✔ , semEmp = 0 ✔ ;

int countA = 1 ✖ , countB = 0;

```
void prodA() {
  semFull.P;
  semA.P;
  //produce an element
  countA -= 2 ✖ ;
  if (countA == 1 ✖ )
  {  countB = 4 ✔ ; semB.V;  }
   else
     semA.V;
  semEmp.V;
}

void prodB() {
  semFull.P;
  semB.P;
  //produce an element
  countB -= 1 ✔ ;
  if (countB == 0 ✔ )
  {  countA = 2 ✔ ; semA.V;  }
```

```
  else
    semB.V;
  semEmp.V;
}
```

Twoja odpowiedź jest częściowo poprawna.

You have correctly selected 8.
The correct answer is:
We have a buffer of capacity N=25, and two processes of producers A and B. Producer A produces the letter A, and producer B produces the letter B.

Complete the code of the prodA and prodB functions, which will be cyclically called by producer A and producer B, respectively.

Each function call concerns 1 element. Semaphores with init, p, v operations should be used.

When using semaphores, it is necessary to ensure synchronization of access to the buffer so that letters are placed in it according to the sequence:

AABBBBAABBBBAABBBBAABBBB .....

```
binsem semA = [1], semB = [0];
sem semFull = [25], semEmp = [0];
int countA = [2], countB = 0;

void prodA() {
  semFull.P;
  semA.P;
  //produce an element
  countA -= [1];
  if (countA == [0])
  {  countB = [4]; semB.V;  }
   else
    semA.V;
  semEmp.V;
}

void prodB() {
  semFull.P;
  semB.P;
  //produce an element
  countB -= [1];
  if (countB == [0])
  {  countA = [2]; semA.V;  }
  else
    semB.V;
  semEmp.V;
}
```

Question **4**

Partially correct

Mark 0.53 out of 1.00

We have a buffer of capacity N=70, and two processes of producers A and B. Producer A produces the letter A, and producer B produces the letter B.

Complete the code of the prodA and prodB functions, which will be cyclically called by producer A and producer B, respectively.

Each function call concerns 1 element. Semaphores with initialization, P and V operations should be used.

When using semaphores, it is necessary to ensure synchronization of access to the buffer so that letters are placed in it according to the sequence:

AABBAABBAABBAABB .....

binsem mutex = 1 ✔ , semA = 1 ✖ ; semB = 0 ✔ ;

```
int count = 0; countA = 0, countB = 3;
bool waitA=false, waitB=false;

void prodA() {
  mutex.P ✔ ;
  if ( (count == N) || (countA == 2)) {
    waitA = true;
    mutex.V ✖ ;
    semB.P ✖ ;
    waitA = false;
  }
  //produce an element
  count += 1;
  countA += 1;
  if (waitB && (count>N) && (countA == 2)) {
    countB=0;
    semA.P ✖ ;
    semA.V ✖ ;
  }
  semB.V ✖ ;
}

void prodB() {
```

mutex.P ✔ ;

if ((count == N) || (countB == 3)) {

waitB = true;

mutex.P ✖ ;

mutex.V ✔ ;

waitB = false;

}

//produce an element

count += 1;

countB += 1;

if (waitA && (count<N) && (countB == 3)) {

countA = 0;

semA.V ✔ ;

mutex.P ✔ ;

}

mutex.V ✔ ;

}

Twoja odpowiedź jest częściowo poprawna.

You have correctly selected 8.

The correct answer is:

We have a buffer of capacity N=70, and two processes of producers A and B. Producer A produces the letter A, and producer B produces the letter B.

Complete the code of the prodA and prodB functions, which will be cyclically called by producer A and producer B, respectively.

Each function call concerns 1 element. Semaphores with initialization, P and V operations should be used.

When using semaphores, it is necessary to ensure synchronization of access to the buffer so that letters are placed in it according to the sequence:

AABBAABBAABBAABB .....

binsem mutex = [1], semA = [0]; semB = [0];

int count = 0; countA = 0, countB = 3;

bool waitA=false, waitB=false;

void prodA() {

[mutex.P];

if ((count == N) || (countA == 2)) {

waitA = true;

[semA.P];

[mutex.V];

```
        waitA = false;
    }
    //produce an element
    count += 1;
    countA += 1;
    if (waitB && (count>N) && (countA == 2)) {
        countB=0;
        [semB.V];
        [mutex.P];
    }
    [mutex.V];
}

void prodB() {
    [mutex.P];
    if ( (count == N) || (countB == 3)) {
        waitB = true;
        [semB.P];
        [mutex.V];
        waitB = false;
    }
    //produce an element
    count += 1;
    countB += 1;
    if (waitA && (count<N) && (countB == 3)) {
        countA = 0;
        [semA.V];
        [mutex.P];
    }
    [mutex.V];
}
```

Question 5

Partially correct

Mark 0.93 out of 1.00

We have a buffer of capacity N=60, and two processes of producers A and B. Producer A produces the letter A, and producer B produces the letter B.

Complete the code of the prodA and prodB functions, which will be cyclically called by producer A and producer B, respectively.

Each function call concerns 1 element. Semaphores with initialization, P and V operations should be used.

When using semaphores, it is necessary to ensure synchronization of access to the buffer so that letters are placed in it according to the sequence:

AABBAABBAABBAABB .....

binsem mutex = 1, semA = 0; semB = 0;

int count = 0; countA = 0 ✔ ; countB=2;

bool waitA = false ✔ , waitB=false;

```
void prodA() {
  mutex.P;
  if ( (count == N) && (countA == 2) ✖ ) {
    waitA = true ✔ ;
    mutex.V;
    semA.P;
    waitA = false ✔ ;
  }
  //produce an element
  count += 1;
  countA += 1 ✔ ;
  if ( waitB && (N>count) && (countA == 2) ✔ ) {
    countB == 0 ✔ ;
    semB.V;
    mutex.P;
  }
  mutex.V;
}

void prodB() {
```

```
  mutex.P;
  if ( (count == N) || (countB == 2) ✔ ) {
    waitB = true ✔ ;
    mutex.V;
    semB.P;
    waitB = false ✔ ;
  }
  //produce an element
  count += 1;
  countB += 1 ✔ ;
  if ( waitA && (N>count) && (countB == 2) ✔ ) {
    countA = 0 ✔ ;
    semA.V;
    mutex.P;
  }
   mutex.V;
}
```

Twoja odpowiedź jest częściowo poprawna.

You have correctly selected 13.

The correct answer is:

We have a buffer of capacity N=60, and two processes of producers A and B. Producer A produces the letter A, and producer B produces the letter B.

Complete the code of the prodA and prodB functions, which will be cyclically called by producer A and producer B, respectively.

Each function call concerns 1 element. Semaphores with initialization, P and V operations should be used.

When using semaphores, it is necessary to ensure synchronization of access to the buffer so that letters are placed in it according to the sequence:

AABBAABBAABBAABB .....

binsem mutex = 1, semA = 0; semB = 0;

int count = 0; [countA = 0]; countB=2;

bool [waitA = false], waitB=false;

```
void prodA() {
  mutex.P;
  if ([(count == N) || (countA == 2)]) {
    [waitA = true];
    mutex.V;
    semA.P;
```

```
        [waitA = false];
    }
    //produce an element
    count += 1;
    [countA += 1];
    if ([waitB && (N>count) && (countA == 2)]) {
        [countB == 0];
        semB.V;
        mutex.P;
    }
    mutex.V;
}

void prodB() {
    mutex.P;
    if ([(count == N) || (countB == 2)]) {
        [waitB = true];
        mutex.V;
        semB.P;
        [waitB = false];
    }
    //produce an element
    count += 1;
    [countB += 1];
    if ([waitA && (N>count) && (countB == 2)]) {
        [countA = 0];
        semA.V;
        mutex.P;
    }
    mutex.V;
}
```

Question 6

Correct

Mark 1.00 out of 1.00

We have a buffer of capacity N=20, and two processes of producers A and B. Producer A produces the letter A, and producer B produces the letter B.

Complete the code of the prodA and prodB functions, which will be cyclically called by producer A and producer B, respectively.

Each function call concerns 1 element. Semaphores with init, p, v operations should be used.

When using semaphores, it is necessary to ensure synchronization of access to the buffer so that letters are placed in it according to the sequence:

AABBAABBAABBAABB .....

binsem semA = 1, semB = 0;

sem semFull = 20, semEmp = 0;

int countA = 0, countB = 0;

```
void prodA() {
  semFull.P ✔ ;
  semA.P ✔ ;
  //produce an element
  countA += 1;
  if (countA == 2)
  {  countB = 0; semB.V ✔ ;  }
  else
    semA.V ✔ ;
  semEmp.V ✔ ;
}

void prodB() {
  semFull.P ✔ ;
  semB.P ✔ ;
  //produce an element
  countB += 1;
  if (countB == 2)
  {  countA = 0; semA.V ✔ ;  }
  else
```

semB.V ✔ ;

semEmp.V ✔ ;

}

Twoja odpowiedź jest poprawna.

The correct answer is:
We have a buffer of capacity N=20, and two processes of producers A and B. Producer A produces the letter A, and producer B produces the letter B.

Complete the code of the prodA and prodB functions, which will be cyclically called by producer A and producer B, respectively.

Each function call concerns 1 element. Semaphores with init, p, v operations should be used.

When using semaphores, it is necessary to ensure synchronization of access to the buffer so that letters are placed in it according to the sequence:

AABBAABBAABBAABB .....

binsem semA = 1, semB = 0;

sem semFull = 20, semEmp = 0;

int countA = 0, countB = 0;

```
void prodA() {
  [semFull.P];
  [semA.P];
  //produce an element
  countA += 1;
  if (countA == 2)
  {   countB = 0; [semB.V];   }
  else
    [semA.V];
  [semEmp.V];
}

void prodB() {
  [semFull.P];
  [semB.P];
  //produce an element
  countB += 1;
  if (countB == 2)
  {   countA = 0; [semA.V];   }
  else
    [semB.V];
  [semEmp.V];
}
```

Question 7

Correct

Mark 1.00 out of 1.00

We have a buffer of capacity N=50, and two processes of producers A and B. Producer A produces the letter A, and producer B produces the letter B.

Complete the code of the prodA and prodB functions, which will be cyclically called by producer A and producer B, respectively.

Each function call concerns 1 element. Semaphores with init, p, v operations should be used.

When using semaphores, it is necessary to ensure synchronization of access to the buffer so that letters are placed in it according to the sequence:

AABBAABBAABBAABB .....

binsem semA = 1, semB = 0;

sem semFull = 50, semEmp = 0;

int countA = 2, countB = 0;

```
void prodA() {
  semFull.P ✔ ;
  semA.P ✔ ;
  //produce an element
  countA -= 1;
  if (countA == 0)
  {  countB = 2; semB.V ✔ ;  }
  else
    semA.V ✔ ;
  semEmp.V ✔ ;
}

void prodB() {
  semFull.P ✔ ;
  semB.P ✔ ;
  //produce an element
  countB -= 1;
  if (countB == 0)
  {  countA = 2; semA.V ✔ ;  }
  else
```

semB.V ✔ ;

semEmp.V ✔ ;

}

Twoja odpowiedź jest poprawna.

The correct answer is:
We have a buffer of capacity N=50, and two processes of producers A and B. Producer A produces the letter A, and producer B produces the letter B.

Complete the code of the prodA and prodB functions, which will be cyclically called by producer A and producer B, respectively.

Each function call concerns 1 element. Semaphores with init, p, v operations should be used.

When using semaphores, it is necessary to ensure synchronization of access to the buffer so that letters are placed in it according to the sequence:

AABBAABBAABBAABB .....

binsem semA = 1, semB = 0;

sem semFull = 50, semEmp = 0;

int countA = 2, countB = 0;

```
void prodA() {
  [semFull.P];
  [semA.P];
  //produce an element
  countA -= 1;
  if (countA == 0)
  {   countB = 2; [semB.V];   }
  else
    [semA.V];
  [semEmp.V];
}

void prodB() {
  [semFull.P];
  [semB.P];
  //produce an element
  countB -= 1;
  if (countB == 0)
  {   countA = 2; [semA.V];   }
  else
    [semB.V];
  [semEmp.V];
}
```

Question 8

Partially correct

Mark 0.92 out of 1.00

We have a buffer of capacity N=30, and two processes of producers A and B. Producer A produces the letter A, and producer B produces the letter B.

Complete the code of the prodA and prodB functions, which will be cyclically called by producer A and producer B, respectively.

Each function call concerns 1 element. Semaphores with init, p, v operations should be used.

When using semaphores, it is necessary to ensure synchronization of access to the buffer so that letters are placed in it according to the sequence:

ABBBABBBABBBABBB .....

binsem semA = 1, semB = 0;

sem semFull = 30, semEmp = 0;

int countA = 0, countB = 0;

```
void prodA() {
  semFull.P ✔ ;
  semA.P ✔ ;
  //produce an element
  countA += 1;
  if ( countA==0 ✖ )
  { countB = 0; semB.V ✔ ; }
  else
    semA.V ✔ ;
  semEmp.V ✔ ;
}

void prodB() {
  semFull.P ✔ ;
  semB.P ✔ ;
  //produce an element
  countB += 1;
  if ( countB==3 ✔ )
  { countA = 0; semA.V ✔ ; }
```

```
  else
    semB.V ✔ ;
  semEmp.V ✔ ;
}
```

Twoja odpowiedź jest częściowo poprawna.

You have correctly selected 11.
The correct answer is:
We have a buffer of capacity N=30, and two processes of producers A and B. Producer A produces the letter A, and producer B produces the letter B.

Complete the code of the prodA and prodB functions, which will be cyclically called by producer A and producer B, respectively.

Each function call concerns 1 element. Semaphores with init, p, v operations should be used.

When using semaphores, it is necessary to ensure synchronization of access to the buffer so that letters are placed in it according to the sequence:

ABBBABBBABBBABBB .....

binsem semA = 1, semB = 0;

sem semFull = 30, semEmp = 0;

int countA = 0, countB = 0;

```
void prodA() {
  [semFull.P];
  [semA.P];
  //produce an element
  countA += 1;
  if ([true])
  {  countB = 0; [semB.V];   }
  else
    [semA.V];
  [semEmp.V];
}

void prodB() {
  [semFull.P];
  [semB.P];
  //produce an element
  countB += 1;
  if ([countB==3])
  {  countA = 0;  [semA.V];   }
  else
    [semB.V];
  [semEmp.V];
}
```

Question 9

Partially correct

Mark 0.92 out of 1.00

We have a buffer of capacity N=45, and two processes of producers A and B. Producer A produces the letter A, and producer B produces the letter B.

Complete the code of the prodA and prodB functions, which will be cyclically called by producer A and producer B, respectively.

Each function call concerns 1 element. Semaphores with init, p, v operations should be used.

When using semaphores, it is necessary to ensure synchronization of access to the buffer so that letters are placed in it according to the sequence:

ABBBABBBABBBABBB .....

binsem semA = 1, semB = 0;

sem semFull = 45, semEmp = 0;

int countA = 1, countB = 0;

```
void prodA() {
  semFull.P ✔ ;
  semA.P ✔ ;
  //produce an element
  countA -= 1;
  if ( countA==1 ✖ )
  { countB = 3; semB.V ✔ ; }
  else
    semA.V ✔ ;
  semEmp.V ✔ ;
}

void prodB() {
  semFull.P ✔ ;
  semB.P ✔ ;
  //produce an element
  countB -= 1;
  if ( countB==0 ✔ )
  { countA = 1; semA.V ✔ ; }
```

```
  else
    semB.V ✔ ;
  semEmp.V ✔ ;
}
```

Twoja odpowiedź jest częściowo poprawna.

You have correctly selected 11.
The correct answer is:
We have a buffer of capacity N=45, and two processes of producers A and B. Producer A produces the letter A, and producer B produces the letter B.

Complete the code of the prodA and prodB functions, which will be cyclically called by producer A and producer B, respectively.

Each function call concerns 1 element. Semaphores with init, p, v operations should be used.

When using semaphores, it is necessary to ensure synchronization of access to the buffer so that letters are placed in it according to the sequence:

ABBBABBBABBBABBB .....

```
binsem semA = 1, semB = 0;
sem semFull = 45, semEmp = 0;
int countA = 1, countB = 0;

void prodA() {
  [semFull.P];
  [semA.P];
  //produce an element
  countA -= 1;
  if ([true])
  {  countB = 3; [semB.V];  }
  else
    [semA.V];
  [semEmp.V];
}

void prodB() {
  [semFull.P];
  [semB.P];
  //produce an element
  countB -= 1;
  if ([countB==0])
  {  countA = 1;  [semA.V];  }
  else
    [semB.V];
  [semEmp.V];
}
```

Question 10

Partially correct

Mark 0.92 out of 1.00

We have a buffer of capacity N=20, and two processes of consumers A and B. Consumer A consumes the letter A, and consumer B consumes the letter B.

Complete the code of the consA and consB functions, which will be cyclically called by consumer A and consumer B, respectively.

Each function call concerns 1 element. Semaphores with initialization, p, v operations should be used.

When using semaphores, it is necessary to ensure synchronization of access to the buffer so that letters are retrieved in it according to the sequence:

BAABAABAABAA .....

binsem semA = 0, semB = 1;

sem semFull = 20, semEmp = 0;

int countA = 0, countB = 0;

```
void consA() {
  semEmp.P ✔ ;
  semA.P ✔ ;
  //consume an element
  countA += 1;
  if ( countA==2 ✔ )
  { countB = 0; semB.V ✔ ; }
  else
    semA.V ✔ ;
  semFull.V ✔ ;
}

void consB() {
  semEmp.P ✔ ;
  semB.P ✔ ;
  //consume an element
  countB += 1;
  if ( countB==2 ✖ )
  { countA = 0; semA.V ✔ ; }
```

```
  else
    semB.V ✔ ;
  semFull.V ✔ ;
}
```

Twoja odpowiedź jest częściowo poprawna.

You have correctly selected 11.
The correct answer is:

We have a buffer of capacity N=20, and two processes of consumers A and B. Consumer A consumes the letter A, and consumer B consumes the letter B.

Complete the code of the consA and consB functions, which will be cyclically called by consumer A and consumer B, respectively.

Each function call concerns 1 element. Semaphores with initialization, p, v operations should be used.

When using semaphores, it is necessary to ensure synchronization of access to the buffer so that letters are retrieved in it according to the sequence:

BAABAABAABAA .....

```
binsem semA = 0, semB = 1;
sem semFull = 20, semEmp = 0;
int countA = 0, countB = 0;

void consA() {
  [semEmp.P];
  [semA.P];
  //consume an element
  countA += 1;
  if ([countA==2])
  {  countB = 0; [semB.V];  }
  else
    [semA.V];
  [semFull.V];
}

void consB() {
  [semEmp.P];
  [semB.P];
  //consume an element
  countB += 1;
  if ([true])
  {  countA = 0; [semA.V];  }
  else
    [semB.V];
  [semFull.V];
}
```

Started on	Tuesday, 6 May 2025, 10:20 AM
State	Finished
Completed on	Tuesday, 6 May 2025, 10:24 AM
Time taken	3 mins 30 secs
Marks	3.00/5.00
Grade	**0.01** out of 0.01 (**60**%)

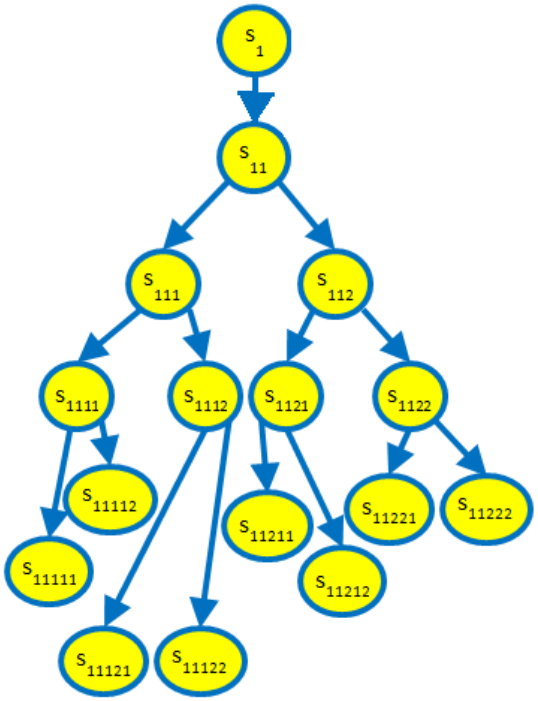
Question **1**

Incorrect

Mark 0.00 out of 1.00

The following figure shows the static nesting structure of subroutines. Subroutine s11211 makes a call to the subroutine that is its "parent" in the hierarchy. To what stack frame (of which subroutine) will the dynamic link be made in the frame of the called subroutine?

Enter the name of the subroutine with the index, without the letter "s", e.g. for s11122 enter 11122



Answer: 1121 ✖

The dynamic link points always to the calling subroutine, so the solution is 11211.

The correct answer is: 11211

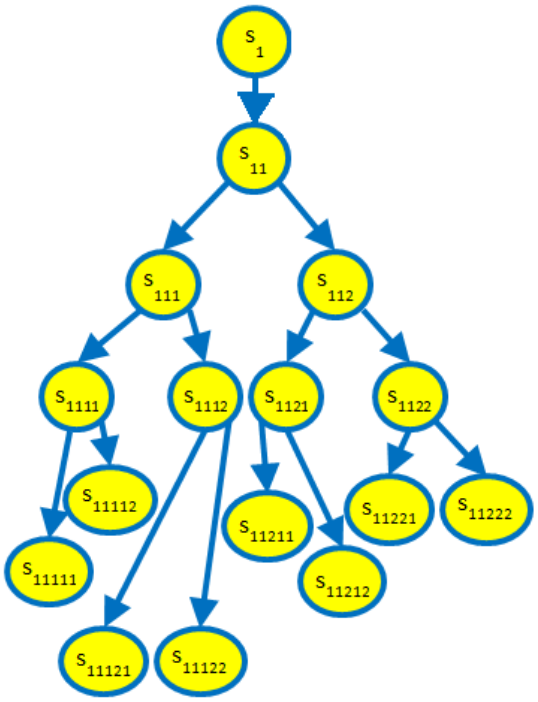
Question 2

Correct

Mark 1.00 out of 1.00

The following figure shows the static nesting structure of subroutines. In the stack frame of the subroutine s11121, can there be a dynamic link to the frame of the subroutine s11222?

answer 0-not, 1-yes



Answer: 0

From a subroutine, only its ancestors and children of its ancestors can be called. So, the subroutine s11222 cannot call the subroutine s11121 and the solution is 0.

The correct answer is: 0

Question 3

Incorrect

Mark 0.00 out of 1.00

The following figure shows the static nesting of subroutines. In the stack frame of the subroutine s112 can there be a dynamic link to the frame of the subroutine s11111?

answer 0-not, 1-yes

Answer: 0 ✖

From a subroutine, only its ancestors and children of its ancestors can be called. So, the subroutine s11111 can call the subroutine s112 and the solution is 1

The correct answer is: 1

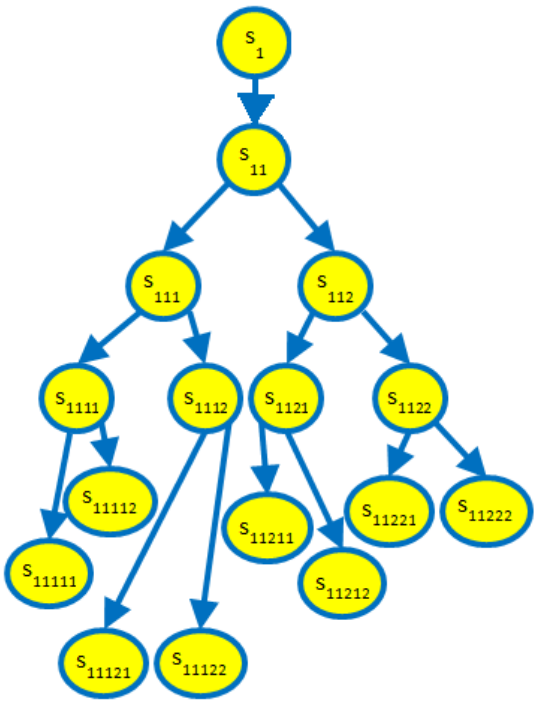
Question **4**

Correct

Mark 1.00 out of 1.00

The figure shows the static nesting of subroutines. In the stack frame of the subroutine s112 can there be a dynamic link to the frame of the subroutine s111?

answer 0-not, 1-yes



Answer: 0

From a subroutine, only its ancestors and children of its ancestors can be called. So, the subroutine s111 cannot call the subroutine s112 and the solution is 0

The correct answer is: 0

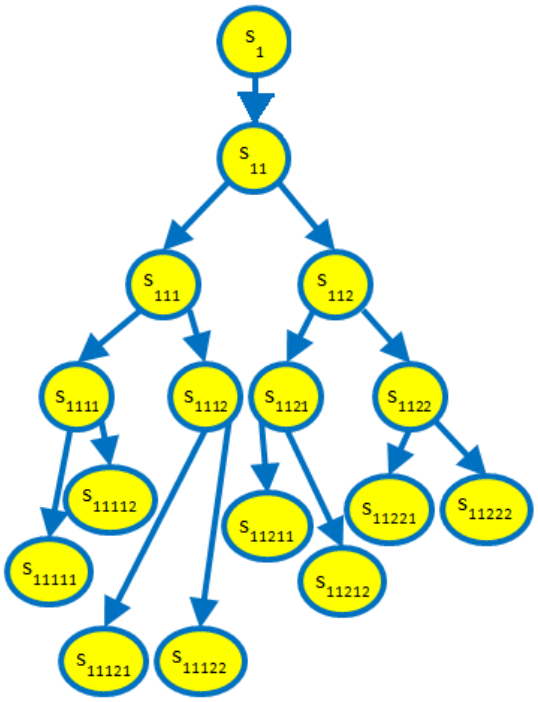
Question 5

Correct

Mark 1.00 out of 1.00

The following figure shows the static nesting structure of subroutines. Subroutine s11 makes a recursive call. To what stack frame (of which subroutine) will the dynamic link be made in the recursively called subroutine frame?

Enter the name of the subroutine with the index, without the letter "s", e.g. for s11122 enter 11122



Answer: 11 ✔

A dynamic points always to the stack frame of the calling subroutine, so in a recursive call it points to the stack frame of the previous incarnation of the same subroutine on the stack. In this case, s11

The correct answer is: 11

Started on	Tuesday, 6 May 2025, 10:30 AM
State	Finished
Completed on	Tuesday, 6 May 2025, 10:33 AM
Time taken	3 mins 13 secs
Marks	1.00/7.00
Grade	**0.00** out of 0.01 (**14.29**%)

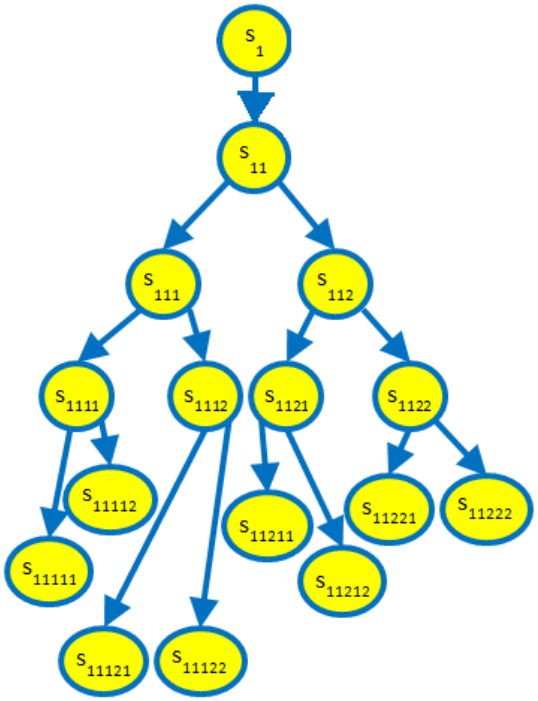
Question **1**

Incorrect

Mark 0.00 out of 1.00

The following graph shows the static nesting structure of subprograms in a particular program.

The subroutine S111 is called from the subroutine S1111. In the frame of the S111 subroutine, the static link will be set to the frame of which subroutine (give the index)? If the call is impossible, enter index -1.



Answer: 1121 ✖

The static link points always to the parent of the called subroutine, so the solution is 11

The correct answer is: 11

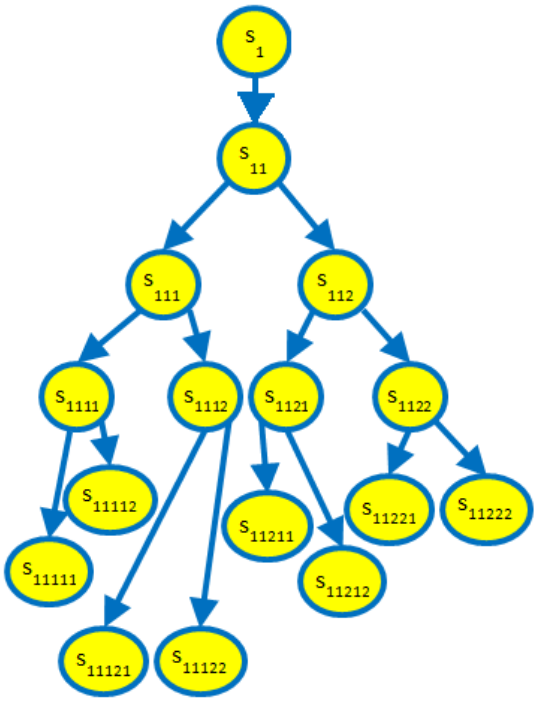
Question 2

Incorrect

Mark 0.00 out of 1.00

The following graph shows the static nesting structure of subprograms in a particular program.

The subroutine S111 is called from the subroutine S11. In the frame of the S111 subroutine, the static link will be set to the frame of which subroutine (give the index)? If the call is impossible, enter index -1.



Answer: 0 ✖

The static link points always to the parent of the called subroutine, so the solution is 11

The correct answer is: 11

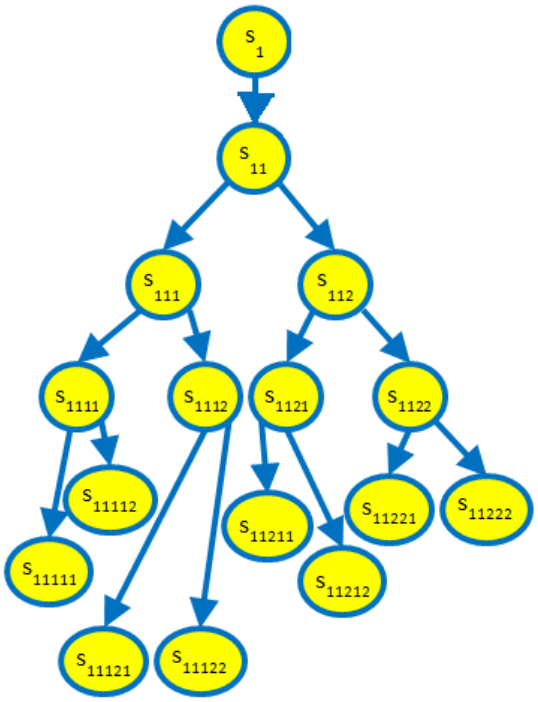
Question 3

Incorrect

Mark 0.00 out of 1.00

The following graph shows the static nesting structure of subprograms in a particular program.

The subroutine S1121 is called from the subroutine S11. In the frame of the S1121 subroutine, the static link will be set to the frame of which subroutine (give the index)? If the call is impossible, enter index -1.



Answer: 0 ✖

The subroutine S1121 cannot be called from the subroutine S11 (only the children, the subroutine recursively, and its ancestors can be called), so the solution is -1

The correct answer is: -1

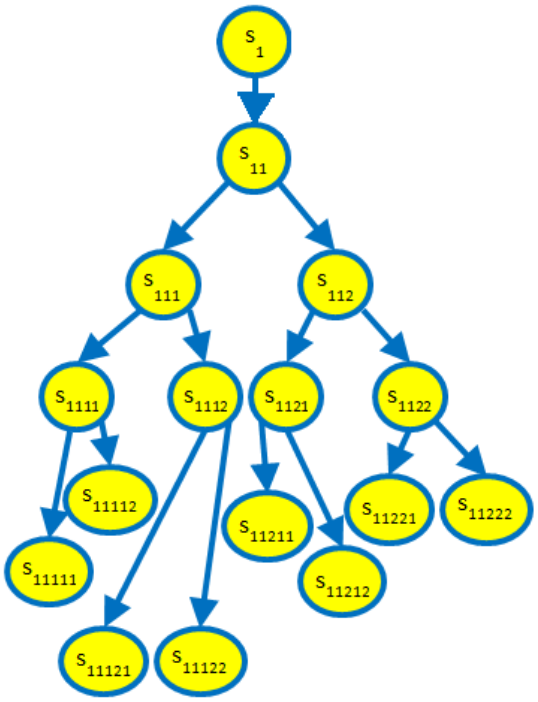
Question **4**

Incorrect

Mark 0.00 out of 1.00

The following graph shows the static nesting structure of subprograms in a particular program.

The subroutine S11 is called from the subroutine S11211. In the frame of the S11 subroutine, the static link will be set to the frame of which subroutine (give the index)? If the call is impossible, enter index -1.



Answer: 11 ✖

The static link points always to the parent of the called subroutine, so the solution is 1

The correct answer is: 1

Question 5

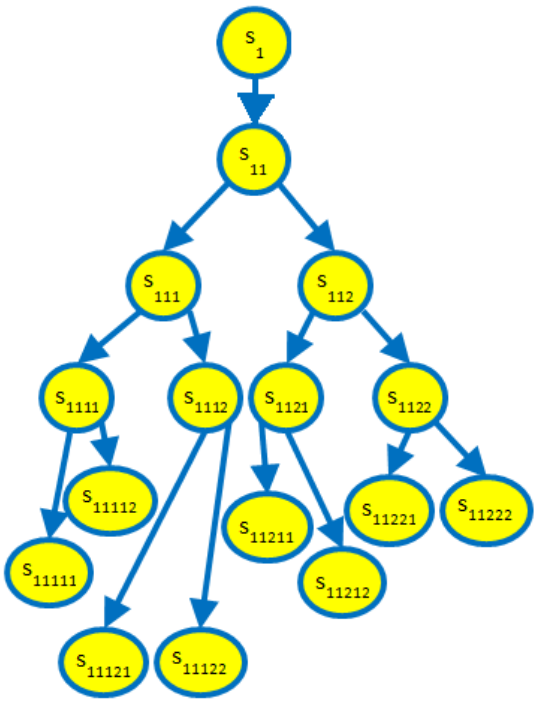
Incorrect

Mark 0.00 out of 1.00

The following graph shows the static nesting structure of subroutines in a particular program.

Can there be in the stack frame of subroutine S11221 a static link to subroutine frame S1122?

answer 0-no, 1-yes



Answer: 0 ✖

The static link points always to the parent of the called subroutine, and the subroutine S1122 is a parent of S11221, so the solution is 1 (yes)

The correct answer is: 1

Question 6

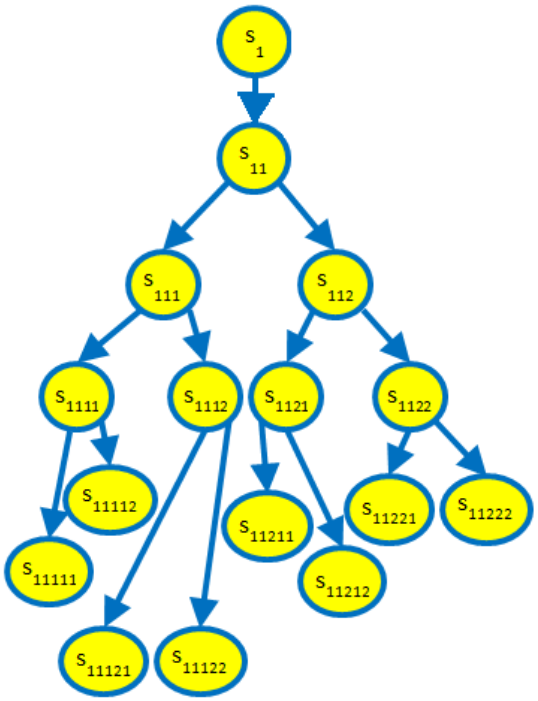
Incorrect

Mark 0.00 out of 1.00

The following graph shows the static nesting structure of subroutines in a particular program.

The subroutine S1122 calls its parent. To which stack frame (of which subroutine) will point the static link in the stack frame of the called subroutine?

Enter the subroutine index, without the letter "S", e.g., for S11122, enter 11122



Answer: 112 ✖

The static link points always to the parent of the called subroutine, and the parent of the subroutine S112 is the subroutine S11, so the solution is 11

The correct answer is: 11

Question **7**

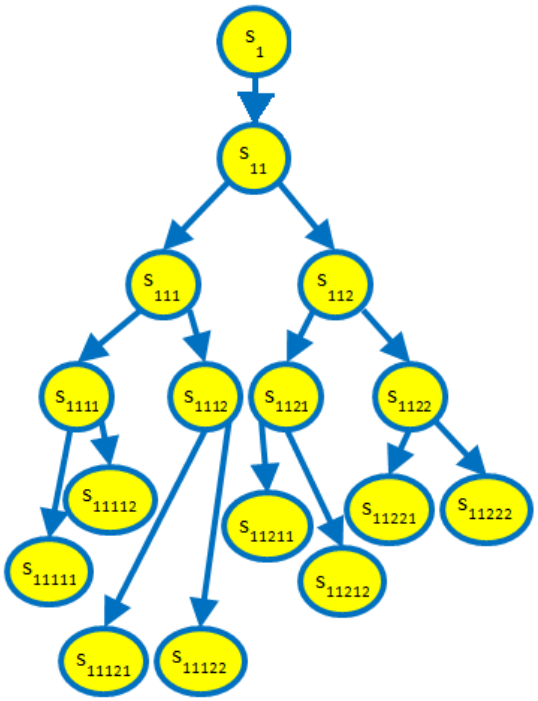
Correct

Mark 1.00 out of 1.00

The following graph shows the static nesting structure of subroutines in a particular program.

The subroutine S1121 makes a recursive call. To which stack frame (of which subroutine) will point the static link in the stack frame of the recursively called subroutine?

Enter the subroutine index, without the letter "S", e.g., for S11122, enter 11122



Answer: 112

The correct answer is: 112

Started on	Tuesday, 6 May 2025, 10:12 AM
State	Finished
Completed on	Tuesday, 6 May 2025, 10:28 AM
Time taken	16 mins 3 secs
Marks	4.00/4.00
Grade	**0.01** out of 0.01 (**100**%)

Question **1**

Correct

Mark 1.00 out of 1.00

What is the average time in the system for tasks in the batch incoming in this order, using FCFS algorithm?

task	1	2
processing time	3.4	5.3

Answer: 6.05 ✔

The first task finishes at time 3.4. The second task finishes at time 3.4+5.3=8.7.

Average = (3.4+8.7)/2=6.05

The correct answer is: 6,05

Question 2

Correct

Mark 1.00 out of 1.00

What is the average time in the system for tasks in the batch incoming in this order, using FCFS algorithm?

We have 2 processors in the system.

task	1	2	3
processing time	3.4	5.3	2.3

Answer: 4.8 ✔

For processor 1, the first task is chosen, and it finishes at time 3.4. For processor 2, the second task is chosen and it finishes at time 5.3.

When task 1 finishes, task 3 is chosen for processor 1 and it finishes at time 3.4+2.3=5.7.

Average = (3.4+5.3+5.7)/3=4.8

The correct answer is: 4,8

Question 3

Correct

Mark 1.00 out of 1.00

What is the average time in the system for tasks in the batch incoming in this order, using SJF algorithm?

task	1	2
processing time	5.3	3.4

Answer: 6.05 ✔

The second task is shorter, so it is chosen first, and it finishes at time 3.4. The first task executes after the second and it finishes at time 5.3+3.4=8.7.

Average = (3.4+8.7)/2=6.05

The correct answer is: 6,05

Question **4**

Correct

Mark 1.00 out of 1.00

What is the average time in the system for tasks in the batch incoming in this order, using SJF algorithm?

We have 2 processors in the system.

task	1	2	3
processing time	3.4	5.3	2.3

Answer: 4.43 ✔

For processor 1, the third task is chosen because it is the shortest one, and it finishes at time 2.3. For processor 2, the first task is chosen because it is the second shortest, and it finishes at time 3.4.

When task 3 finishes, task 2 is chosen for processor 1 and it finishes at time 2.3+5.3=7.6.

Average = (2.3+3.4+7.6)/3=4.43

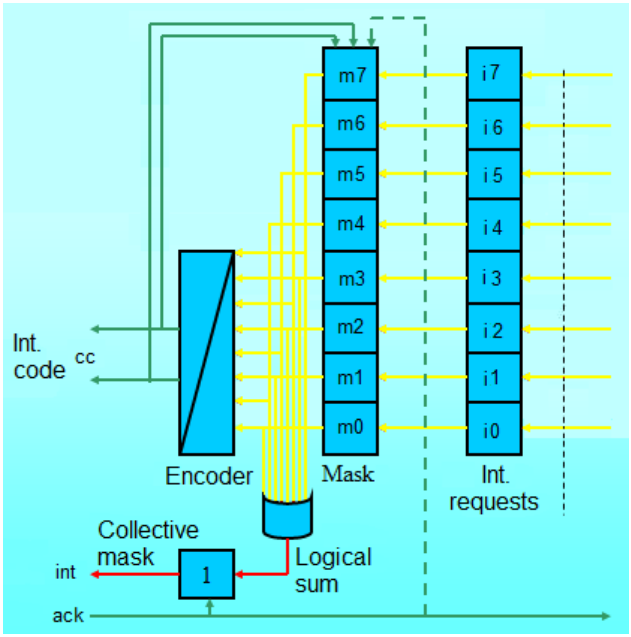
The correct answer is: 4,43

Started on	Tuesday, 6 May 2025, 10:15 AM
State	Finished
Completed on	Tuesday, 6 May 2025, 10:24 AM
Time taken	9 mins 33 secs
Marks	1.00/4.00
Grade	**0.00** out of 0.01 (**25**%)

Question **1**

Correct

Mark 1.00 out of 1.00



In the given interrupt controller structure, the interrupt mask is 11111000 (from m7 to m0), and interrupts 01011010 (from i7 to i0) are reported.

The interrupt with index 7 has the highest priority.

What will be the value of the interrupt code cc sent to the processor (provide a decimal value)?

If no interrupt is reported, enter -1

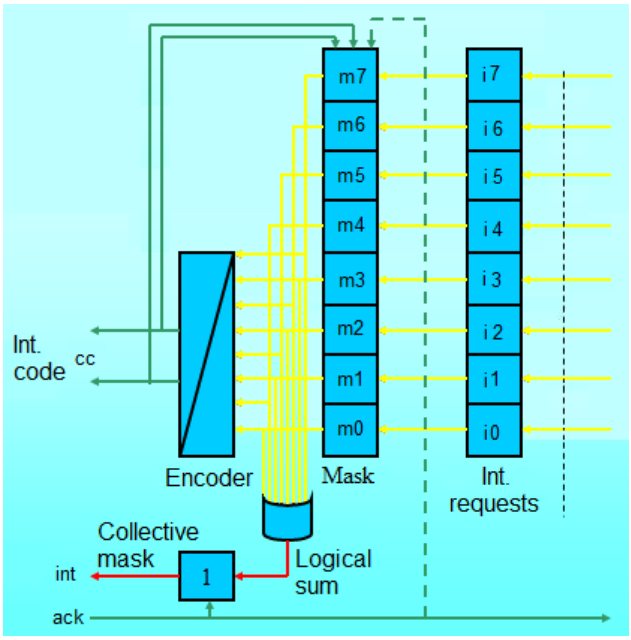
Answer: 6 ✔

The correct answer is: 6

Question 2

Incorrect

Mark 0.00 out of 1.00



In the given interrupt controller structure, the interrupt mask is 11100000 (from m7 to m0). Enter the interrupt number from 0 to 7, without the prefix "i", that caused this mask form.

The interrupt with index 7 has the highest priority.

If no interrupt could cause this mask form, enter -1.

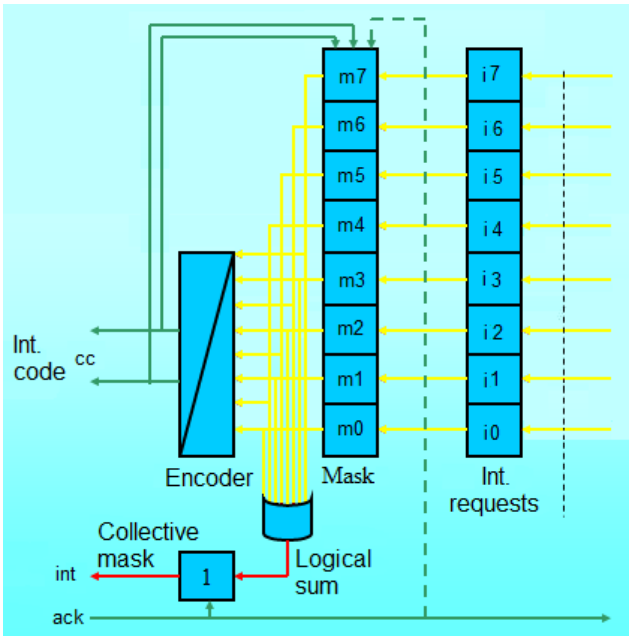
Answer: 5 ✖

The correct answer is: 4

Question 3

Incorrect

Mark 0.00 out of 1.00



In the given interrupt controller structure, the interrupt mask is 11111000 (from m7 to m0), and interrupts 00101100 (from i7 to i0) are reported.

The interrupt with index 7 has the highest priority.

What will be the new value of the interrupt mask? Provide the bits m7m6m5m4m3m2m1m0, for example: 01010101

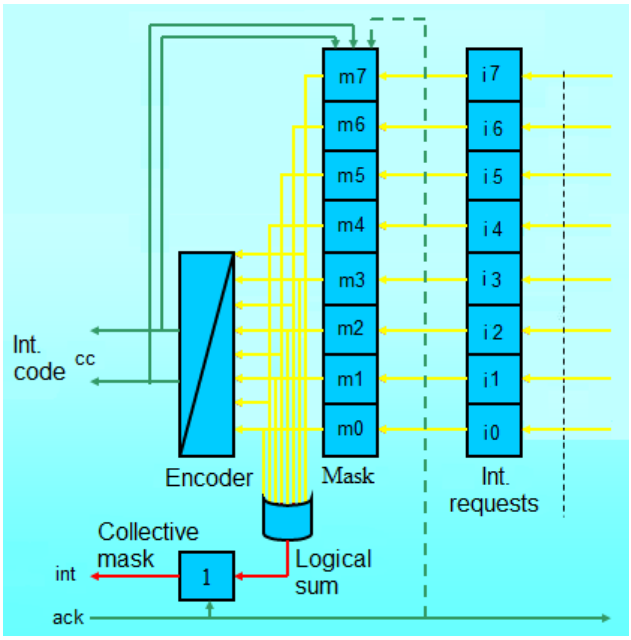
Answer: 11100000 ✖

The correct answer is: 11000000

Question **4**

Incorrect

Mark 0.00 out of 1.00



In the given interrupt controller structure, the interrupt number cc reported to the processor is equal to 5. What will be the form of the interrupt mask (from m7 to m0) after the processor confirms the reception of this interrupt?

The interrupt with index 7 has the highest priority.

Enter the mask in the form m7m6m5m4m3m2m1m0, for example: 01010101

If this is not possible, enter -1.

Answer: 11100000 ✖

The correct answer is: 11000000

Started on	Tuesday, 6 May 2025, 10:36 AM
State	Finished
Completed on	Tuesday, 6 May 2025, 10:53 AM
Time taken	17 mins 3 secs
Marks	6.00/22.00
Grade	**0.00** out of 0.01 (**27.27**%)

Question **1**

Correct

Mark 1.00 out of 1.00

Call parameters
Static link
Result
Trace
Dynamic link
Local variables and work fields

The stack frame is shown above. If the result is an automatic dynamic variable pointed to by the address, what memory area does the result field point to? stack ✔

Twoja odpowiedź jest poprawna.

The correct answer is:

Call parameters
Static link
Result
Trace
Dynamic link
Local variables and work fields

The stack frame is shown above. If the result is an automatic dynamic variable pointed to by the address, what memory area does the result field point to? [stack]

Question 2

Correct

Mark 1.00 out of 1.00

Call parameters
Static link
Result
Trace
Dynamic link
Local variables and work fields

The stack frame is shown above. If the result is a dynamically managed variable pointed to by address, what memory area does the result field point to? heap ✔

Twoja odpowiedź jest poprawna.

The correct answer is:

Call parameters
Static link
Result
Trace
Dynamic link
Local variables and work fields

The stack frame is shown above. If the result is a dynamically managed variable pointed to by address, what memory area does the result field point to?[heap]

Question 3

Correct

Mark 1.00 out of 1.00

Call parameters
Static link
Result
Trace
Dynamic link
Local variables and work fields

The stack frame is shown above. If the result is a dynamically allocated variable pointed to by address, what memory area does the result field point to? heap ✔

Twoja odpowiedź jest poprawna.

The correct answer is:

Call parameters
Static link
Result
Trace
Dynamic link
Local variables and work fields

The stack frame is shown above. If the result is a dynamically allocated variable pointed to by address, what memory area does the result field point to?[heap]

Question **4**

Correct

Mark 1.00 out of 1.00

Call parameters
Static link
Result
Trace
Dynamic link
Local variables and work fields

The stack frame is shown above. If the result is a global variable, pointed to by address, what area of memory does the result field point to?

static data ✔

Twoja odpowiedź jest poprawna.

The correct answer is:

Call parameters
Static link
Result
Trace
Dynamic link
Local variables and work fields

The stack frame is shown above. If the result is a global variable, pointed to by address, what area of memory does the result field point to? [static data]

Question **5**

Correct

Mark 1.00 out of 1.00

Call parameters
Static link
Result
Trace
Dynamic link
Local variables and work fields

The stack frame is shown above. What memory area does the trace point to? code ✔

Twoja odpowiedź jest poprawna.

The correct answer is:

Call parameters
Static link
Result
Trace
Dynamic link
Local variables and work fields

The stack frame is shown above. What memory area does the trace point to? [code]

Question 6

Incorrect

Mark 0.00 out of 1.00

The organization of memory and processor is word-base (int also takes one 16-bit word)

variable a has the value #a0c0

top of the stack (full descending) #c100

stack frame before calling subroutine A #c105

address of subroutine A #1001

Calling rules: C convention, no static link, function result passed in registers. The stack is shown after subroutine A is called, at label point C. Subroutine A is called A(a,a,a); from address #1050.

Regardless of the programming language, subroutine A has the form:

```
A(int x,y,z);
{
        int q = x+0x20;
C: ...
}
```

Specify, in hexadecimal, what value the cell with the address #c0ff will contain.

Address	content	
#c100	????	
#c0ff		
#c0fe	#a0c0	
#c0fd	#a0c0	
#c0fc	#1051	
#c0fb	#c105	
#c0fa	#a0e0	
#c0f9	????	
#c0f8	????	

Answer: a0e0 ✖

The correct answer is: #a0c0

Question **7**

Incorrect

Mark 0.00 out of 1.00

The organization of memory and processor is word-base (int also takes one 16-bit word)

variable a has the value #a0c0

top of the stack (full descending) #c100

stack frame before calling subroutine A #c105

address of subroutine A #1001

Calling rules: C convention, no static link, function result passed in registers. The stack is shown after subroutine A is called, at label point C. Subroutine A is called A(a,a,a); from address #1050.

Regardless of the programming language, subroutine A has the form:

```
A(int x,y,z);
{
        int q = x+0x20;
C: ...
}
```

Specify, in hexadecimal, what value the cell with the address #c0fe will contain.

Address	content	
#c100	????	
#c0ff	#a0c0	
#c0fe		
#c0fd	#a0c0	
#c0fc	#1051	
#c0fb	#c105	
#c0fa	#a0e0	
#c0f9	????	
#c0f8	????	

Answer: a0c0 ✖

The correct answer is: #a0c0

Question 8

Correct

Mark 1.00 out of 1.00

The organization of memory and processor is word-base (int also takes one 16-bit word)

variable a has the value #a0c0

top of the stack (full descending) #c100

stack frame before calling subroutine A #c105

address of subroutine A #1001

Calling rules: C convention, no static link, function result passed in registers. The stack is shown after subroutine A is called, at label point C. Subroutine A is called A(a,a,a); from address #1050.

Regardless of the programming language, subroutine A has the form:

```
A(int x,y,z);
{
        int q = x+0x20;
C: ...
}
```

Specify, in hexadecimal, what value the cell with the address #c0fd will contain.

Address	content	
#c100	????	
#c0ff	#a0c0	
#c0fe	#a0c0	
#c0fd		
#c0fc	#1051	
#c0fb	#c105	
#c0fa	#a0e0	
#c0f9	????	
#c0f8	????	

Answer: #a0c0 ✔

The correct answer is: #a0c0

Question 9

Incorrect

Mark 0.00 out of 1.00

The organization of memory and processor is word-base (int also takes one 16-bit word)

variable a has the value #a0c0

top of the stack (full descending) #c100

stack frame before calling subroutine A #c105

address of subroutine A #1001

Calling rules: C convention, no static link, function result passed in registers. The stack is shown after subroutine A is called, at label point C. Subroutine A is called A(a,a,a); from address #1050.

Regardless of the programming language, subroutine A has the form:

```
A(int x,y,z);
{
        int q = x+0x20;
C: ...
}
```

Specify, in hexadecimal, what value the cell with the address #c0fc will contain.

Address	content	
#c100	????	
#c0ff	#a0c0	
#c0fe	#a0c0	
#c0fd	#a0c0	
#c0fc		
#c0fb	#c105	
#c0fa	#a0e0	
#c0f9	????	
#c0f8	????	

Answer: 1051 ✖

The correct answer is: #1051

Question 10

Incorrect

Mark 0.00 out of 1.00

The organization of memory and processor is word-base (int also takes one 16-bit word)

variable a has the value #a0c0

top of the stack (full descending) #c100

stack frame before calling subroutine A #c105

address of subroutine A #1001

Calling rules: C convention, no static link, function result passed in registers. The stack is shown after subroutine A is called, at label point C. Subroutine A is called A(a,a,a); from address #1050.

Regardless of the programming language, subroutine A has the form:

```
A(int x,y,z);
{
        int q = x+0x20;
C: ...
}
```

Specify, in hexadecimal, what value the cell with the address #c0fb will contain.

Address	content	
#c100	????	
#c0ff	#a0c0	
#c0fe	#a0c0	
#c0fd	#a0c0	
#c0fc	#1051	
#c0fb		
#c0fa	#a0e0	
#c0f9	????	
#c0f8	????	

Answer: c105 ✖

The correct answer is: #c105

Question **11**

Incorrect

Mark 0.00 out of 1.00

The organization of memory and processor is word-base (int also takes one 16-bit word)

variable a has the value #a0c0

top of the stack (full descending) #c100

stack frame before calling subroutine A #c105

address of subroutine A #1001

Calling rules: C convention, no static link, function result passed in registers. The stack is shown after subroutine A is called, at label point C. Subroutine A is called A(a,a,a); from address #1050.

Regardless of the programming language, subroutine A has the form:

```
A(int x,y,z);
{
        int q = x+0x20;
C: ...
}
```

Specify, in hexadecimal, what value the cell with the address #c0fa will contain.

Address	content	
#c100	????	
#c0ff	#a0c0	
#c0fe	#a0c0	
#c0fd	#a0c0	
#c0fc	#1051	
#c0fb	#c105	
#c0fa		
#c0f9	????	
#c0f8	????	

Answer: a0e0 ✖

The correct answer is: #a0e0

Question 12

Incorrect

Mark 0.00 out of 1.00

The organization of memory and processor is word-based (int also takes 1 word)

variable a has the value #c0a0

top of the stack (full descending) #c100

address of subroutine A #1010

Calling rules: C convention, no static link, function result passed in registers. The contents of the stack after preparing the subroutine call parameters A. Subprogram A is called A(a,a,a); from address #10a0.

Regardless of the programming language, the subroutine header A has the form:

A(int x,y,z);

Specify, in hexadecimal, what value the cell with the address #c0ff will contain.

Adres	zawartość	
#c100	????	
#c0ff		
#c0fe	#a0a0	
#c0fd	#c0a0	
#c0fc	????	
#c0fb	????	

Answer: c0a0 ✖

The correct answer is: #a5a0

Question 13

Incorrect

Mark 0.00 out of 1.00

The organization of memory and processor is word-based (int also takes 1 word)

variable a has the value #c0a0

top of the stack (full descending) #c100

address of subroutine A #1010

Calling rules: C convention, no static link, function result passed in registers. The contents of the stack after preparing the subroutine call parameters A. Subprogram A is called A(a,a,a); from address #10a0.

Regardless of the programming language, the subroutine header A has the form:

A(int x,y,z);

Specify, in hexadecimal, what value the cell with the address #c0fe will contain.

Adres	zawartość	
#c100	????	
#c0ff	#a5a0	
#c0fe		
#c0fd	#c0a0	
#c0fc	????	
#c0fb	????	

Answer: c0a0 ✖

The correct answer is: #a0a0

Question **14**

Incorrect

Mark 0.00 out of 1.00

The organization of memory and processor is word-based (int also takes 1 word)

variable a has the value #c0a0

top of the stack (full descending) #c100

address of subroutine A #1010

Calling rules: C convention, no static link, function result passed in registers. The contents of the stack after preparing the subroutine call parameters A. Subprogram A is called A(a,a,a); from address #10a0.

Regardless of the programming language, the subroutine header A has the form:

A(int x,y,z);

Specify, in hexadecimal, what value the cell with the address #c0fd will contain.

Adres	zawartość	
#c100	????	
#c0ff	#a5a0	
#c0fe	#a0a0	
#c0fd		
#c0fc	????	
#c0fb	????	

Answer: c0a0 ✖

The correct answer is: #c0a0

Question 15

Incorrect

Mark 0.00 out of 1.00

The organization of memory and processor is word-based (int also takes 1 word)

Variable a has a value #a0a0

Stack top (full descending) #c100

stack frame before calling the subroutine B #c01d

subroutine B address #10d1

Rules for calling: Language C convention, without a static link, the result of the function returned in the registers. The content of the stack is shown after calling the subroutine B, and then recursive by itself, at the label C. Subroutine B is called for the first time B(a); somewhere in the program from the #1cff address. The label C has #1d10 address

Regardless of the programming language, the subroutine B is in the form

```
B (int i);
{
  int j=i-2;
C: B (j-1);
}
```

What is the value, in hexadecimal, of the cell at address #c0ff?

Address	content	
#c100	????	
#c0ff		
#c0fe	#1d00	
#c0fd	#c01d	
#c0fc	#a09e	
#c0fb	#a09d	
#c0fa	#1d11	
#c0f9	#c0fd	
#c0f8	#a09b	
#c0f7	????	

Answer: a09f ✖

The correct answer is: #a0a0

Question 16

Incorrect

Mark 0.00 out of 1.00

The organization of memory and processor is word-based (int also takes 1 word)

Variable a has a value #a0a0

Stack top (full descending) #c100

stack frame before calling the subroutine B #c01d

subroutine B address #10d1

Rules for calling: Language C convention, without a static link, the result of the function returned in the registers. The content of the stack is shown after calling the subroutine B, and then recursive by itself, at the label C. Subroutine B is called for the first time B(a); somewhere in the program from the #1cff address. The label C has #1d10 address

Regardless of the programming language, the subroutine B is in the form

```
B (int i);
{
  int j=i-2
C: B (j-2);
}
```

What is the value, in hexadecimal, of the cell at address #c0fe?

Address	content	
#c100	????	
#c0ff	#a0a0	
#c0fe		
#c0fd	#c01d	
#c0fc	#a09e	
#c0fb	#a09c	
#c0fa	#1d11	
#c0f9	#c0fd	
#c0f8	#a09a	
#c0f7	????	

Answer: a0a0 ✖

The correct answer is: #1d00

Question **17**

Incorrect

Mark 0.00 out of 1.00

The organization of memory and processor is word-based (int also takes 1 word)

Variable a has a value #a0a0

Stack top (full descending) #c100

stack frame before calling the subroutine B #c01d

subroutine B address #10d1

Rules for calling: Language C convention, without a static link, the result of the function returned in the registers. The content of the stack is shown after calling the subroutine B, and then recursive by itself, at the label C. Subroutine B is called for the first time B(a); somewhere in the program from the #1cff address. The label C has #1d10 address

Regardless of the programming language, the subroutine B is in the form

```
B (int i);
{
  int j=i-1;
C: B (j-2);
}
```

What is the value, in hexadecimal, of the cell at address #c0fd?

Address	content	
#c100	????	
#c0ff	#a0a0	
#c0fe	#1d00	
#c0fd		
#c0fc	#a09f	
#c0fb	#a09d	
#c0fa	#1d11	
#c0f9	#c0fd	
#c0f8	#a09c	
#c0f7	????	

Answer: c01d ✖

The correct answer is: #c01d

Question 18

Incorrect

Mark 0.00 out of 1.00

The organization of memory and processor is word-based (int also takes 1 word)

Variable a has a value #a0a0

Stack top (full descending) #c100

stack frame before calling the subroutine B #c01d

subroutine B address #10d1

Rules for calling: Language C convention, without a static link, the result of the function returned in the registers. The content of the stack is shown after calling the subroutine B, and then recursive by itself, at the label C. Subroutine B is called for the first time B(a); somewhere in the program from the #1cff address. The label C has #1d10 address

Regardless of the programming language, the subroutine B is in the form

```
B (int i);
{
   int j=i-2;
C: B (j-2);
}
```

What is the value, in hexadecimal, of the cell at address #c0fc?

Address	content	
#c100	????	
#c0ff	#a0a0	
#c0fe	#1d00	
#c0fd	#c01d	
#c0fc		
#c0fb	#a09c	
#c0fa	#1d11	
#c0f9	#c0fd	
#c0f8	#a09a	
#c0f7	????	

Answer: a09f ✖

The correct answer is: #a09e

Question 19

Incorrect

Mark 0.00 out of 1.00

The organization of memory and processor is word-based (int also takes 1 word)

Variable a has a value #a0a0

Stack top (full descending) #c100

stack frame before calling the subroutine B #c01d

subroutine B address #10d1

Rules for calling: Language C convention, without a static link, the result of the function returned in the registers. The content of the stack is shown after calling the subroutine B, and then recursive by itself, at the label C. Subroutine B is called for the first time B(a); somewhere in the program from the #1cff address. The label C has #1d10 address

Regardless of the programming language, the subroutine B is in the form

```
B (int i);
{
  int j=i-1;
C: B (j-2);
}
```

What is the value, in hexadecimal, of the cell at address #c0fb?

Address	content	
#c100	????	
#c0ff	#a0a0	
#c0fe	#1d00	
#c0fd	#c01d	
#c0fc	#a09f	
#c0fb		
#c0fa	#1d11	
#c0f9	#c0fd	
#c0f8	#a09c	
#c0f7	????	

Answer: a09d ✖

The correct answer is: #a09d

Question 20

Incorrect

Mark 0.00 out of 1.00

The organization of memory and processor is word-based (int also takes 1 word)

Variable a has a value #a0a0

Stack top (full descending) #c100

stack frame before calling the subroutine B #c01d

subroutine B address #10d1

Rules for calling: Language C convention, without a static link, the result of the function returned in the registers. The content of the stack is shown after calling the subroutine B, and then recursive by itself, at the label C. Subroutine B is called for the first time B(a); somewhere in the program from the #1cff address. The label C has #1d10 address

Regardless of the programming language, the subroutine B is in the form

```
B (int i);
{
  int j=i-2;
C: B (j);
}
```

What is the value, in hexadecimal, of the cell at address #c0fa?

Address	content	
#c100	????	
#c0ff	#a0a0	
#c0fe	#1d00	
#c0fd	#c01d	
#c0fc	#a09e	
#c0fb	#a09e	
#c0fa		
#c0f9	#c0fd	
#c0f8	#a09c	
#c0f7	????	

Answer: 1d10 ✖

The correct answer is: #1d11

Question 21

Incorrect

Mark 0.00 out of 1.00

The organization of memory and processor is word-based (int also takes 1 word)

Variable a has a value #a0a0

Stack top (full descending) #c100

stack frame before calling the subroutine B #c01d

subroutine B address #10d1

Rules for calling: Language C convention, without a static link, the result of the function returned in the registers. The content of the stack is shown after calling the subroutine B, and then recursive by itself, at the label C. Subroutine B is called for the first time B(a); somewhere in the program from the #1cff address. The label C has #1d10 address

Regardless of the programming language, the subroutine B is in the form

```
B (int i);
{
  int j=i-1;
C: B (j-1);
}
```

What is the value, in hexadecimal, of the cell at address #c0f9?

Address	content	
#c100	????	
#c0ff	#a0a0	
#c0fe	#1d00	
#c0fd	#c01d	
#c0fc	#a09f	
#c0fb	#a09e	
#c0fa	#1d11	
#c0f9		
#c0f8	#a09d	
#c0f7	????	

Answer: c0fd ✖

The correct answer is: #c0fd

Question 22

Incorrect

Mark 0.00 out of 1.00

The organization of memory and processor is word-based (int also takes 1 word)

Variable a has a value #a0a0

Stack top (full descending) #c100

stack frame before calling the subroutine B #c01d

subroutine B address #10d1

Rules for calling: Language C convention, without a static link, the result of the function returned in the registers. The content of the stack is shown after calling the subroutine B, and then recursive by itself, at the label C. Subroutine B is called for the first time B(a); somewhere in the program from the #1cff address. The label C has #1d10 address

Regardless of the programming language, the subroutine B is in the form

```
B (int i);
{
  int j=i-3;
C: B (j-2);
}
```

What is the value, in hexadecimal, of the cell at address #c0f8?

Address	content	
#c100	????	
#c0ff	#a0a0	
#c0fe	#1d00	
#c0fd	#c01d	
#c0fc	#a09d	
#c0fb	#a09b	
#c0fa	#1d11	
#c0f9	#c0fd	
#c0f8		
#c0f7	????	

Answer: a098 ✖

The correct answer is: #a098

1. The system stack must provide space for:
- Processor registers in as many copies as there are interrupt lines +1

2. Twice consecutive execution in one process of operation P on a raised binary semaphore:
- Causes the semaphore to get down
- If a semaphore guards a critical region, it leads to a deadlock

3. In UNIX, access rights are attributes:
- An entry in the I-node table

4. The kernel (microkernel) is responsible for:
- interrupt handling (at the elementary level, then they are passed on to other layers)
- synchronization of processes and devices with processes
- task control

5. In multi-level interrupts (vectored interrupts) the jump is performed:
- according to the table indexed by the interrupt number

6. In UNIX, the disk contains the following areas:
- Files data
- I-node table
- SuperBlock

7. What is this technique where the operating system gathers programs and data together before processing?
- Batch processing

8. The page fault interrupt is used to:
- Downloads to the memory of the requested page

9. Scheduling algorithms can be:
- Indulgent
- Preemptive

10. The resources of the computer system are:
- Primary memory
- Peripheral devices
- Processor time

11. When is the scheduler called?
- at the end of the execution of each kernel procedure

12. A memory management technique in which the system divides memory into equal-sized portions to easily manage relocation is called:
- paging

13. What does the file system layer do?
- Manages files
- Manages directories
- Manages free storage space
- Tracks the status of information

14. Indulgent scheduling is the best mechanism for:
- real-time system

15. What does the kernel do when there is no task (process) to run?
- Starts the idle task

16. Discontinuous allocation is the result of:
- segmentation
- paging

17. Paging is in thrashing if:

- the system spends more time paging than execution

18. The return from interrupt instruction:
- restores the conditions register
- restores the program counter

19. Dynamic relocation requires the use of:
- Base register (DATUM)

20. A program compiled for execution in paged memory can be executed in regular memory.
False

21. Page thrashing is a phenomenon involving:
- frequent downloading of pages that have just been swapped out from memory
- Frequent loading of pages that have just been ejected from memory

22. Semaphores are used to solve the problem:
- Mutual exclusion

23. Dirty frame is:
- modified

24. The common allocation queue to fixed blocks of memory of equal size causes:
- Internal fragmentation

25. Which scheduler needs to make a decision the fastest?
- short-term

26. In operating system:
- Every exception must be handled

27. Static relocation is performed by:
- compiler
- linker
- Loader

28. Operation V on a raised binary semaphore:
- It does not change the value of the semaphore

29. Fragmentation is implied by:
- Internal by paging
- Internal by partitioning into quantized blocks
- External by freeing blocks in a deallocation order not reverse to allocation
- External by resizing allocated blocks

30. A process <u>can</u> appear in the pool of scheduling processes as a result of:
- Performing V operation on the semaphore
- Completing an I/O operation
- Starting a new process

31. External fragmentation <u>can be avoided</u> by:
- Allocation of fixed size static memory blocks
- Freeing memory blocks in the reverse order to allocation

32. The Test-And-Set or Compare-And-Swap instruction has the following characteristics:
- Requires active waiting from tasks
- It is possible to use only on computers with common memory
- It is possible to deadlock with TAS/CAS operations on multiple variables

33. Address translation is handled by a unit called (give the abbreviation):
- MMU

34. Dynamic relocation:
- Requires hardware support in the form of a DATUM register

- Allows for temporal elimination of external fragmentation

35. Opening a file in UNIX writes the following entries in the operating system's data structures:
- Inserting a new entry into the Table of Active I-nodes or increasing the counter in an existing entry
- Inserting a new item into the Table of Open Files of the Process

36. What type of code can multiple processes execute simultaneously?
- reentrant

37. Address translation aims to:
- Converting a virtual address to a physical one

38. The optimal scheduling algorithm in terms of minimizing the average time in the system of a given task is:
- SJF

39. Inter-process communication can be organized using:
- messages
- shared directly addressable memory fields
- shared memory fields accessible through system calls

40. For concurrency in OS:
- Interrupt handling is necessary

41. Which of the following information is stored on task switching?
- I/O status information
- scheduler data
- contents of general purpose registers, program counter, and similar registers available to the program
- Contents of datum, limit and other registers inaccessible to the program

42. The conversion of the effective address to the physical one takes place:
- In the memory management unit

43. Which of the following statements is true for system level threads?
- Kernel-level threads require their descriptors in the kernel

44. Address translation mechanism:
- Concatenates the frame number and offset on the page

45. Dynamic relocation is performed by:
- Paging system
- Special registers (DATUM)
- Segment descriptors

46. What is included in the context that must be maintained for a synchronous precision interrupt?
- general purpose registers
- program counter
- collective or individual interrupt mask

47. System/user threads:
- User-level thread descriptors are stored in the address space of the program
- User-level threads share the same execution context
- System level thread descriptors are stored in the operating system kernel

48. How is exception identification performed?
- the specification of hardware interrupts is given over the data bus

49. The combination of paging and segmentation consists in:
- the use of a segment table or a pool of segment registers treated as an additional, superior level of paging

50. The behavior of the exchange algorithm opposite to that expected with the measures taken is called:
- anomaly

51. The hardware resources of a computer system are:
- Processor time
- Primary memory
- Peripheral devices

52. How does the operating system call the task completion subroutine?
1. → builds the frame of the terminating subroutine on the task stack
2. → sets the trace in the terminating subroutine to the current position
3. → builds an interrupt vector on the system stack pointing to the terminating subroutine code
4. → recreates the context programmatically and executes the IRET instruction

53. What mechanism is part of time-sharing systems?
- short-term scheduler
- low-level scheduler

54. On UNIX, the number of files is directly limited by:
- I-node table size
- The size of the space allocated for files

55. What are the functions of the kernel?
- Interrupt handling

56. What type of code can be executed simultaneously on multiple processors?
- reentrant

57. MMU uses index tables to:
- generating a physical address
- generate a physical address

58. To end the interrupt service, use the following instruction:
- Special return instruction

59. Physical address:
- points to a location in the address space of primary memory

60. Threads in operating system:
- They share context except registers and stack

61. The page fault exception is specific in that:
- is reported in the "middle" of an instruction execution
- execution of the instruction may require decrementing the program counter
- continuation of the instruction execution may require the saving of internal processor registers storing intermediate values

62. Virtual memory consists of:
- Primary memory and storage memory
- primary and mass memory

63. The scheduling goal, which is to occupy processors as efficiently as possible, is:
- utilization

64. The use of timer interrupts is necessary:
- In multiuser systems
- At the suspended state is a process that:
- Waits for an I/O operation to complete

65. Which of the following memory allocation schemes causes external fragmentation?

- Segmentation
- Sweeping
- Multiple contiguous fixed partitions of various sizes

66. Which of the following interrupts a running process?
- Hardware interrupt
- Timer interrupts
- Power fail interrupt

67. Conditional variables in a monitor:
- They are used to suspend processes that cannot run because the conditions for their continuation are not met

68. In a FAT-based disk system (without sharing allocation units by files), the number of files is directly limited by:
- FAT table size
- The size of the disk space

69. In a FAT-based disk system, file size is directly limited by:
- The size of the disk space
- The number of bits of the field describing the size of the file

70. The scheduler decisions take the form:
- change from ready to active state

71. Interrupt vector is saved in a case of:
- accepting a hardware interrupt
- accepting a non-maskable interrupt
- jump with trace

72. Allocation unit for file storage:
- must be constant across the disk partition
- may vary between partitions

73. Allocation unit for storing files:
- must be constant across the disk partition
- may vary between partitions

74. FIRST-FIT algorithm:
- Causes external fragmentation

75. Processor access scheduling decisions may be made under which of the following circumstances?
- When a task goes from the active state to the waiting state
- When a task transitions from the active state to the ready state
- When a task goes from the waiting state to the ready state
- When a task terminates

76. The following situations cause the exceptions (processor internal interrupts) of the "error" type:
- a reference to memory that is not in the address space
- memory reference beyond limit register value
- an attempt to write to the page for which the "read only" bit is set

77. Multiprogram systems:
- It holds more than one program in primary memory at the same time

78. The direct resume rule means that:
- The resuming process loses the critical region
- The resuming process applies for the critical region just like other processes on monitor input

- The resuming process applies for the critical region just like other processes waiting to enter the critical region

79. What is true for simultaneous execution in the same context?
- Threads minimize context switch time
- The use of threads ensures concurrency within the process
- A multiprocessor kernel can be concurrent

80. The following situations trigger "error" exceptions:
- attempting to execute an illegal instruction
- attempting to execute in user mode an instruction that is only legal in system mode
- a reference to memory that is not in the program address space
- memory reference beyond limit register value
- an attempt to execute an instruction from the area of the page for which the "no code" bit was set

81. When an exception is raised in user mode, the operating system switches to the kernel system stack, and what happens when an exception is raised in system mode?
- nothing special, it builds the context on the kernel system stack

82. The number of condition variables is <u>by definition</u> in the monitor:
- as many as there are different conditions for the continuation of processes

83. What mechanism is used to desynchronize processes with different relative speeds?
- buffer

84. Is IOW bit:
- Protection bit

85. Sequence of actions when starting a new task:
1. filling in the descriptor in the kernel
2. memory allocation if this is the first task of the program
3. initialize the stack, fill the first frame
4. instruction to transfer control to the task

86. A multiprogram system is one that, <u>in principle</u>:
- It allows for storing many programs in primary memory
- It allows for storing many programs in the main memory
- Requires relocation or equivalent mechanism

87. A single-program system is one that, in principle:
- Allows for storing only one program in primary memory

88. The common allocation queue to fixed blocks of memory of different sizes causes:
- internal fragmentation
- external fragmentation

89. If the compiler prepares a program with absolute addresses to be loaded in a fixed address space, it is called:
- Static relocation

90. During the interrupt handling:
- other interrupts may or may not be accepted at the discretion of the programmer

91. Scheduling aims to optimize:
- processor utilization
- system throughput
- wait time
- reaction time

92. The page index table address is stored in:

- page table base register

93. Page Index Table address is kept in:
- page table base register

94. The file system layer plays the following role in the operating system:
- Performs file opening and closing operations
- It performs directory services in the hierarchy of disk files
- Controls file access rights

95. The algorithm in which the most recently loaded page is swapped out is called (enter the abbreviation if applicable):
- FIFO

96. The algorithm in which the most recently used page is swapped out is called (enter the abbreviation if applicable):
- LRU

97. In which swapping algorithms is the M bit value used?
- NRU

98. Using Test-And-Set or Compare-And-Swap in synchronization:
- It requires processes to actively wait
- It can only be applied in systems with shared memory

99. Using the Test-And-Set instruction in synchronization:
- It requires processes to actively wait

100. Memory partitioning into blocks of any size:
- It does not imply fragmentation

101. What can happen when a job leaves the critical section and more than 1 task is waiting for the critical section?
- starvation awaiting a critical section

102. The root directory of the disk should be located:
- in a place designated by the data structure in a fixed location on the disk

103. Which technique was introduced because a single task could occupy both the CPU and I/O devices?
- Interrupts

104. Individual interrupt mask:
- a register whose bits are ANDed (conjunction) with the bits from the interrupt request register

105. Which swapping algorithms use the current time slice number?
- Second chance
- Working set

106. Paging-two-level index tables mode of operation
- The content of a level 1 table element points to a level 2 table

107. What is a scheduler?
- a kernel routine that selects a task to execute

108. In indulgent scheduling, the process keeps the CPU until:
- waiving
- termination

109. With indulgent scheduling, once a CPU is allocated to a task, the task keeps it until:
- releasing the processor by the task
- task termination

110. Two successive executions of operation V in one process on a binary semaphore in down state:
- If the semaphore guards a critical region, it can lets two processes enter the critical region

- Raises the semaphore if there are no suspended processes

111. The operating system plays the following role in a computer system:
- Create a concurrent environment
- computer system resource management
- creating abstraction (virtualization) of hardware

112. What mechanism is used to save and restore the task state?
- Context switch

113. What mechanism is used to preserve the states of preempted tasks?
- Context switch

114. Internal fragmentation consists in:
- The program does not use all the memory allocated to it

115. The following paging exceptions allow for returning to the program after they occur:
- page fault
- interrupt on write (IOW)

116. For address translation, the following is used:
- index table
- associative translation buffer

117. In UNIX, access rights are specified:
- Individually for each file
- Separately for the user, the group to which the user belongs and for all others
- Separately for write, read and execute/search

118. Calling the program with the "exec" operation results in (accurately to the result of the exec operation):
- Loading code and data segments from disk, initializing a new stack segment

119. Scheduler function is:
- selecting a task to run

120. In the philosophers problem, if there are 4 of them:
- Deadlock is possible
- Starvation is not possible

121. The software resources of a computer system are:
- Files
- Buffers
- Semaphores

122. Can the exchange algorithm: selecting a frame to empty and loading a new page, be performed in user mode?
- True

123. Interrupt mask:
- causes interrupts to be disabled when an interrupt is accepted
- can unblock interrupts when the interrupt handler completes
- can be set and reset by special processor instructions

124. Which swapping algorithms use page reference history?
- LRU
- LFU

125. The thread is also called:
- Lightweight process

126. Round-robin scheduling is the best mechanism for:
- time-sharing system

127. Priority scheduling is the best mechanism for:
- system with many process classes

128. Context switch is:
- writing registers to the task stack and retrieving them from another task stack

129. Separate allocation queues for fixed memory blocks of equal size cause:
- internal fragmentation
- external fragmentation

130. A multiuser system is one that, in principle:
- Must be concurrent
- Requires the use of timer interrupts

131. In which swapping algorithms is it necessary to collectively clear the M-bits?
- No algorithm
- NRU (? w pdf jest inaczej niż w docx)

132. Interrupt request register is:
- a register where interrupt line states are stored

133. The internal state of the file system layer is available for:
- OS kernel
- Program supervisor layer

134. "Soft" real-time system:
- Guarantees average response time

135. The key in associative memory is:
- Page number

136. The sweeping mechanism consists in:
- Loading segments into primary memory and sending them to disk

137. The interrupt acceptance sequence consists of (in the sequence):
- identification of the interrupt level, saving the interrupt vector, performing a jump with a trace according to the interrupt table

138. Imprecise interrupts can be handled:
- after clearing the pipeline from the instructions
- after saving the full state of the pipeline

139. Logical address:
- points to a cell in the process address space
- is converted to a physical address in the address translation mechanism

140. Imprecise interrupts are:
- accepted in any state of the processor, not just between the execution of successive instructions

141. Precise interrupts are:
- accepted only in a stable state between the execution of successive instructions

142. Internal fragmentation <u>can be</u> removed by:
- No response from the others

143. The mechanism for moving programs between primary memory and mass storage is called:
- sweeping

144. Semaphore function is to:
- Synchronize critical resources to prevent a deadlock

145. Between fork and exec operations, the following operations are performed:
- Opening the appropriate input/output files

146. The sweeping criteria <u>include</u>:
- Priority

- Program state
- Analysis of program execution history

147. The number of tasks performed on the system in a given time is:
- throughput

148. By definition, a deadlock is a situation where:
- any greater than zero number of processes are waiting for conditions that cannot be met

149. SJF selects the task:
- with the least CPU requirement

150. Which swapping algorithms use the history of page references (when was it last used or in which time slices was it used)?
- second chance
- LRU
- LFU
- working set

151. The page reference and modification bits are used to:
- counting references for swapping optimization
- counting references for optimize exchanges

152. When the processor is released, the scheduler selects one of the queued processes:
- ready

153. The page error (page fault) interrupt is used to:
- download the requested page to the primary memory

154. Logical address is:
- effective address

155. The logical address is also:
- Effective address

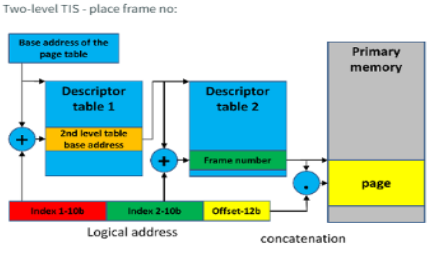
156. In what states can a task occur?
- current
- ready
- blocked

157. The effective address is at the same time:
- logical address

158. The effective address is also:
- Logical address

159. The content of the last (lowest in the hierarchy) page index table is:
- frame

160. Two-level TIS - place frame no:



- in the table of the second level - on the right in the figure

161. The environment in which the process is executed includes:
- A set of environment variables

- Process address space
- General purpose registers content
- Open files

162. Which of the following applies to user-level threads?
- User-level threads cost no execution time in system mode

163. WORST-FIT algorithm:
- Requires a descending sort of the list of free blocks
- It allows for fast determining whether there is a free block of the required size
- It is designed to reduce external fragmentation
- Requires sorting the cut part into the list of free blocks

164. The hardware mechanisms necessary for paging are ("frame error" also called "page fault", "frame miss"):
- address translation, page index tables, "frame error" interrupt

165. Having two-level page index tables:
- the content of a level I table element points to a level II table

166. Which swapping algorithms use information about a reference to pages in the last k periods of time?
- working set
- working set clock

167. When a suspended program is moved to auxiliary memory, its process state is called:
- Swept away

168. Which of the following facilities or abilities are required to provide mutual exclusion support?
- A task that is performed outside the critical section must not affect the behavior of a task in the critical section.
- The task stays in its critical section only for a finite amount of time

169. If a linker prepares a program with absolute addresses to be loaded in a fixed address space, it is called:
- Static relocation

170. At the blocked state is a process that:
- waits for an I/O operation to complete

171. Collective interrupt mask is:
- a register that blocks or unblocks all interrupts

172. Which swapping algorithms can be implemented based on hardware support in the form of a collective reading of reference bits and collective clearing of these bits?
- LRU
- LFU

173. In a multitasking environment, the operating system decides which task the CPU gets, when, and for how long. This feature is called:
- Task scheduling

174. Is memory protection useless in a non-concurrent system?
- False

175. The program supervisor layer in the operating system has the following role:
- Intercepts all program system calls and routes them to the appropriate layers
- Runs programs
- Deals with the management of primary memory (memory allocation to programs)

176. A process always transitions from the "user" state to the "system" state as a result of:
- A software interrupt that calls a system function

- Debugger trap

177. At the ready state is a process that:
- Waits for a processor

178. The kernel is _____ user-level threads.
- unaware

179. After performing the mount(/dev/hd5,/usr/x/bin/hd5) operation, the file /usr/z/a on the mounted disk should be referenced by:
- /usr/x/bin/hd5/usr/z/a

180. In a concurrent environment, the operating system decides which task the CPU gets, when, and for how long. This feature is called:
- Task scheduling

181. There are three processes in the system:
C - calculation process (batch process performing complex calculations lasting several hours),
T - text editor (user edits text document)
K - compiler (the user compiles the program, e.g. in C++).
Assign processes to priorities in the operating system from the highest to the lowest priority.
- highest [T]
- intermediate [K]
- lowest [C]

182. In which swapping algorithms is it necessary to clear the M bits individually?
- No algorithm

183. In round-robin scheduling, if each task is allotted a certain amount of time to execute, it is called:
- Time slice

184. The program must be specially compiled to run in paged memory.
- false

185. Interrupt vector:
- it is saved automatically when an interrupt is accepted
- contains minimal information that cannot be saved programmatically

186. Paging - the key in associative memory is:
- Page number

187. Frame protection bits:
- Must be available for writing

188. The multi-level interrupt controller includes:
- Collective interrupt mask
- Interrupt request register
- Individual interrupt mask
- Priority encoder

189. Compaction solves the problem:
- External fragmentation

190. Converting the effective address to physical is performed in:
- memory management unit

191. The microkernel of the operating system performs the following role:
- Synchronizes processes
- It receives interrupts and routes them to the appropriate drivers and other layers of the system

192. Consider the following sequence of address references:
123, 215, 600, 1234, 76, 96.
If the page size is 100, the order of page references is as follows:
- 1,2,6,12,0,0

(simply divide by 100, discard remainders, if 0.87 or sth, it is 0)

193. Multiprogramming is a technique in which, as a rule:
- many programs can be stored in primary memory

194. What is included in the context that must be saved for a synchronous (inter-instruction) precision interrupt?
- collective of individual interrupt mask
- general purpose registers
- program counter

195. General semaphore:
- is to perform only indivisible operations
- is a shared variable
- can be only non-negative

196. What is true about system level threads?
- All process threads share the same address space.
- All process threads can share the same set of open files.
- All process threads can share the same set of child processes.

197. In the case of hardware interrupt summation ("wire or"):
- the processor can programatically poll devices about issuing an interrupt
- the bus driver can poll devices about issuing an interrupt

198. Multi-threading on a multi-processor machine:
- increases concurrency

199. Essential activity in the interrupt handling procedure (i.e., the activity for which the interrupt is issued) consists in:
- unlocking the process waiting for this interrupt

200. Which task queue can never be empty?
- running tasks

201. The process context includes:
- General purpose registers
- Process descriptor (? – question 79 & 59)
- Code and data

202. When starting a program, how is control passed to it from the operating system?
- return from interrupt handler IRET

203. Which of the following statements applies to the process?
- A process is defined as a set of resources needed to run a program.
- The execution of the process must proceed in a sequential manner.
- A process is a running program.

204. The purpose of mutual exclusion is:
- obtaining exclusive access

205. Inter-process communication can be organized using:
- shared memory fields accessible through system calls
- shared directly addressable memory fields
- messages

206. What does it mean that the interrupt subsystem is vectored (all components of the correct answer must be given)?
- interrupts are accepted on multiple input lines
- there is an interrupt handling table indexed by the interrupt line number
- there is an individual interrupt mask

207. The interrupt encoder is:
- a combinational circuit that computes the number of the reported and unmasked interrupt with the highest priority
- a combinational circuit that calculates the value of a new individual interrupt mask
- a combinational circuit that transmits to the processor the number of the interrupt to be serviced

208. Which scheduling is used to organize concurrency?
- short-term

209. What mechanism is part of batch systems?
- medium-term scheduler
- long-term scheduler

210. What are the sequence of actions in interrupt handling?
1. saving a copy of the interrupt vector (PC and SR)
2. switching to system mode (modification of PC and status register)
3. programmatic context saving
4. switching to the system stack

211. After accepting an interrupt, the next interrupts are:
- blocked

212. Which scheduler is also called a job planner?
- long-term

213. What exception (software or hardware interrupt) causes system mode if user mode is current?
- any exception

214. In a multiprocessor operating system, interrupt blocking is sufficient to prevent the microkernel from executing its routines simultaneously.
- False

215. The difference between trap and error is:
- errors are generally reported asynchronously and traps synchronously
- after an error, there is usually no return to the program, and after a trap, yes

216. In the "current" state, there is a process that:
- occupies a processor

217. Devices report their readiness by:
- issuing an interrupt
- setting a status bit

218. Context switch is caused by:
- Interrupts

219. The result of cooperation of concurrent processes:
- it can be non-deterministic
- it may depend on how processes are scheduled

220. The following situations cause "error" exceptions (processor internal interrupts):
- illegal instruction
- instruction legal but prohibited in user mode

221. Cloning a process with a fork operation results in (not taking to account the numerical result of fork):

- Duplication of data segment and stack segment

222. The interrupt vector consists of:

- program counter
- status register
- condition bits

223. Which mechanisms are supported by the phenomenon of locality of references?

- page swapping
- multilevel page index tables
- reverse page index tables
- associative memory of page references

224. Scheduling disc access involves deciding on the following:

- The order in which disc access requests should be handled

225. A typical collection of program segments includes (come of them can be combined):

- Code segment
- Data segment
- Stack segment

226. The relocating loader generates addresses as the program is loaded into primary memory. These addresses are:

- Absolute