Homework 4

Question 1:

a.

Logical Address	Physical Address
The address generated by the CPU	The address actually seen by the memory hardware

b.

	Contiguous storage	Non-contiguous storage
1	A program occupies a single contiguous block of storage locations.	A program is divided into several blocks or segments that may be placed in memory in pieces not necessary adjacent to one another.
2	Faster in store	Slower in store
3	More waste in memory	Less waste in memory

c.

	First-fit	Best-fit
1	The incoming job is placed in the first hole that is big enough.	The incoming job is placed in the small hole that is big enough.
2	It does not need search the entire hole set to place the job.	It need search the entire hole set to place the job.
3	It is faster to allocate.	It is slower.
4	More waste in storage	Less waste in storage

Question 2:

First-fit = 17
Best-fit = 17
Next-fit = 25
Worst-fit = 25

Question 3:

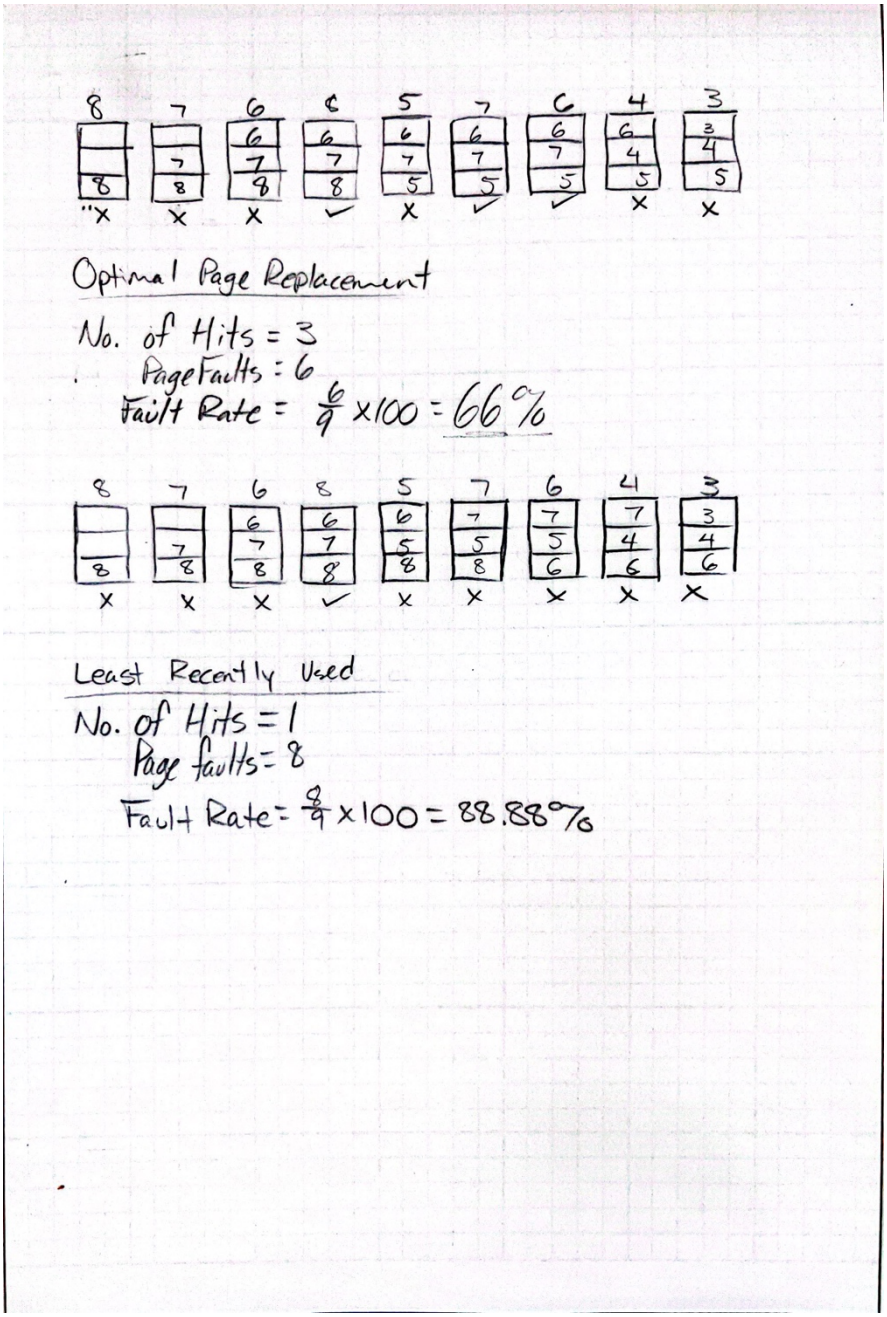
a. It allows allocating to run a program that its size larger than the amount of memory.
b. To solve the fragmentation problem in segmentation storage.

Question 4:

Logical Address			Page Table			Physical Address	
Page No.	Content		Page no.	Frame No.		Free Frame No.	Content
1	aa,bb,cc,dd		1	6		6	aa,bb,cc,dd
2	ee,ff,gg,hh		2	7		7	ee,ff,gg,hh
3	ii,jj,kk,ll		3	10		10	ii,jj,kk,ll
4	mm,nn,oo,pp		4	12		12	mm,nn,oo,pp
5	qq,rr,ss,tt		5	18		18	qq,rr,ss,tt
6	uu,vv,ww,xx		6	20		20	uu,vv,ww,xx
7	yy,zz		7	21		21	yy,zz
						25	
						26	
						29	

a. Allocated each page to free frames list
b. Logical and physical address mapping is shown in page table
c. Page table is constructed to show the mapping of each page to free frame
d. Physical address for instructions = Frame Number*Page_Size+Offset
   bb – 6*4+1 = 25
   ff – 7*4+1 = 29
   rr – 18*4+1 = 73
   vv – 20*4 +1 = 81
e. There exist internal fragmentation. We allotted a page of size 4 bytes to frame 21. But we have only two bytes occupied in it. So there is 2 bytes Internal Fragmentation that exists.

Question 5



Optimal Page Replacement

No. of Hits = 3

Page Faults = 6

Fault Rate = $\frac{6}{9} \times 100 = 66\%$

Least Recently Used

No. of Hits = 1

Page faults = 8

Fault Rate = $\frac{8}{9} \times 100 = 88.88\%$

Question 6:

1. No this won't help since the CPU is under-utilized. This will likely have no effect. The limiting factor is available memory per program.
2. No, this won't help because the problem is speed of retrieval not the amount of room needed.
3. No, this will mean still fewer frames per process and more thrashing making the problem worse. This typically decreases CPU utilization because less memory is available to each program and the chances of page faults increase.
4. Yes, this will help. Fewer process, more frames per process and less thrashing. This typically increases CPU utilization by keeping more of the working set of each program in memory, thereby reducing the number of page faults.
5. Yes, this will help. More frames in memory so less thrashing.
6. No. Since each process would have fewer frames available and the page fault rate would increase

Question 7:

1. When there are no pages which could be used by process is present in memory then it will generate page fault. Initially when process starts execution, we can characterize the page-fault rate equal to number of page fault occurred divided by number of instructions executed. Hence more the number of instructions present per page, lesser will be page fault rate.
2. Once the entire working set of pages is loaded, then page fault can occur only if number of pages require to execute the process cannot be accommodated into physical memory at the same time. Hence the page-fault rate will be equal to number of page fault occurred divided by number of pages accessed by the process. Hence page-fault rate will be less if a greater number of pages are accessed with less number of page fault.

Question 8:

One of the option is to use virtual memory in which logical memory space is larger than primary memory and hence some part of program not currently being used by process resides in the secondary memory and hence based on demand, the pages from secondary memory can be fetched to primary memory.

Another thing is to use global page replacement policy where process can replace the frames used by other process and hence this process can use more physical memory space on demand if other processes in the system does not require much space.