Cameron Bourque UIN: 826004886

Design:

I structured the ContFramePool to have a bitmap as an unsigned char to minimize space needed to hold the frame pool. I set the free_frames count, base_frame_no, n_frames, info_frame_no and n_info_frames to be unsigned long so that they can count to higher numbers or addresses when required.

I made a linked list for the frame pools by making first and last pointers to ContFramePools to get access to the list. This allows for the release frames function to find the correct frame pool and release the frames. There are also next and prev pointers which allow for the linked list to connect frame pools together. These are set to public so they can be accessed from the release frames function which is static since it needs them to iterate through the linked list.

I added a non static release frame sequence function which is called from the static version so that it can use the private member variables such as the bitmap in order for the chosen frame pool to release the frames in a simple manner.

I added #defines called FREE, ALLOCATED, and HEAD_OF_SEQUENCE to define the state of a frame to make the code a little easier to read and prevent errors due to mixing up values of states in my code.

When constructing the frame pool, I make sure to attach it to the end of linked list first. I set all private variables with their corresponding passed in variables. Then I allocate the bitmap at the location determined. Then we loop through all the frames and free them all. If we were given the info frame location to be at 0 then we allocate the info frame in the bitmap. I also check that values are valid for the frame pool.

For get frames, I loop through each bitmap index and check to see if any frames are free. If so then I loop to make sure the required number of frames are free in sequence. If that works out then I allocate the sequence of frames and return the head of the sequence. Otherwise I make sure to check the next frame and keep searching. I make sure to subtract the number of allocated frames from the free frames counter. I also check that all values are valid for the frame pool. If there are no available frames to allocate then I return 0.

For mark inaccessible, I loop from the start frame and allocate the sequence of frames following after marking the start frame as head of sequence. I make sure to subtract the number of frames allocated from the free frames counter. I also make sure all values are in valid range for the frame pool.

For releasing frames, I loop through the linked list starting from the first frame pool and check if the frame to release is in the pool. If not then I check the next frames until there are none left. If none have the frame then I throw an error. If the pool has the frame then I release the head of sequence frame. If the frame isn't head of sequence then I throw an error. Otherwise, I loop through all the frames after the head of sequence frame and, as long as they are marked allocated, I release them.

For the needed info frames function, I returned the solution to the equation the determines how many info frames are needed to manage a given number of frames. If the quotient has a remainder then I round up so that it can handle all the frames.