One advanced data structure that the team used during a competition was a triple ended queue. Which is pretty neat and simple to implement. The code also was relatively trivial. First, we need to be able to push\_front, push\_middle, and push\_back, to do so the solution that we came up with was to use two deques in c++ and basically treat them as one long queue. Additionally, we wanted our two deques to be balanced for performance and we did so by keeping their sizes as close to the same as possible.

The two deques were called f and b for front and back.

Our first command was push\_front

if (command == "push\_front"){

f.emplace\_front(x);

if (f.size() > b.size()+1){

b.emplace\_front(f.at(f.size()-1));

f.pop\_back();

}

}

This logic is straight forward as long as f is smaller than b just place onto the front of f, but if it is larger by 2 or more then move whatever is in the back of f to the front of b.

The next command is push\_back

else if (command == "push\_back"){

b.emplace\_back(x);

if (b.size() > f.size()+1){

f.emplace\_back(b.at(0));

b.pop\_front();

}

}

The logic is nearly identical to push\_front here just in the opposite direction.

The third command is push\_middle

else if (command == "push\_middle"){

int y = f.size() + b.size();

y = (y+1)/2;

if (y < f.size()){

b.emplace\_front(x);

}else{

f.emplace\_back(x);

}

}

Here the main trouble is making sure that the two deques stay balanced and making sure you are placing the object into the right deque. To do this, we basically take the average of the two deque sizes to find the center and if the center falls in the front of the deque then we place the object in the front of the back deque. Likewise, if the center is in the back of the deque then we place the object in the back of the front deque.

The last command is get

else if (command == "get"){

if (x >= f.size()){

cout << b.at(x-f.size()) << "\n";

}else{

cout << f.at(x) << "\n";

}

}

This one was pretty easy, if the location is within the size of the f deque then get the element at that position. If the location is in the back deque then get whatever object is in location x – size of the font deque. Obviously this function only works if the insert functions above were properly formatted.