EGCI 211 Advanced Computer Programming
Dr.Mingmanas Sivaraksa

Mahidol University
International College

# Surgical Readiness Reporting System

By Akira Panyawongkhanti 6381306
Burin Intachuen 6480250
Natthaphat Teanchai 6381322
Tanakorn Mankhetwit 6480282

# Table of content

Page

# User Manual

Code repository
https://github.com/Strike32/medicalreport

There are 2 ways to start our program

1) type g++ medrep.cpp -o medrep, then ./medrep followed by "-i" for inserting mode and "-q" for queuing mode

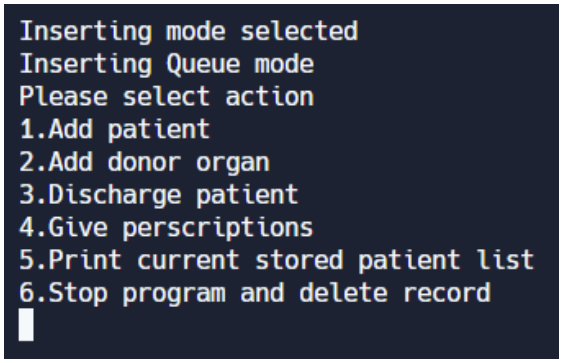
2) type

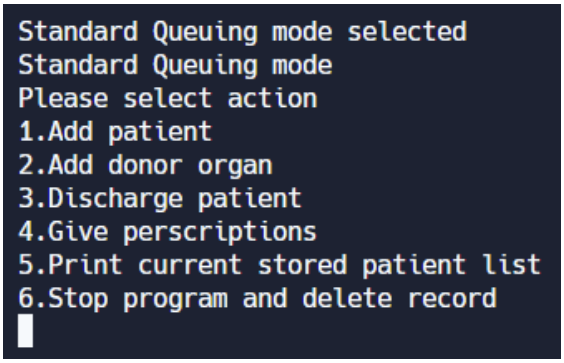
make compile inserting for inserting mode

Or

Make compile queuing for queuing mode

The program will let you know if your mode selection happened properly and which mode had been selected.




Selecting

1) Add patient, would prompt user to type a patient's informations.

2) Add donor organ, would prompt user to type a donor organ information.

3) Discharge patient, would discharge a patient from the list of patients.

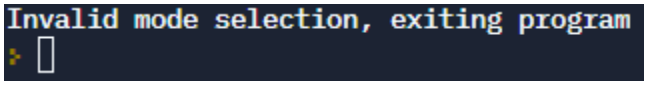
4) Give prescriptions, would allow users to give prescriptions to patients, correct prescriptions would reduce said patient's risk factor and wrong prescriptions would increase his/her risk factor.

5) Print current stored patient list, would ask the user if they want the printed list to be in Ascending or Descending order and would print in selected order.

6) Stop program and delete record, would stop the program as well as delete patients and organs from linked list.

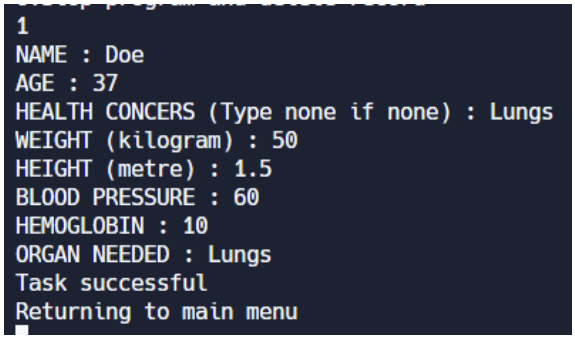
Once selected task is executed, the system would print task successful and return to the main menu again.

From the very start, if mode selection was typed with any other parameters other than “-i” or “-q”, the program would exit with this prompt on-screen.

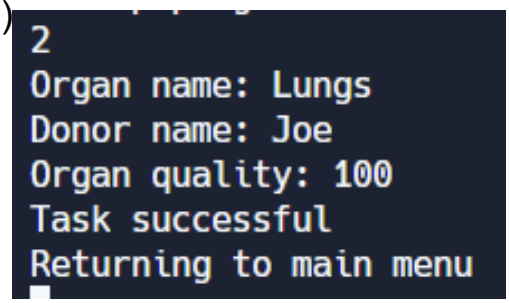



**Demo of selections**

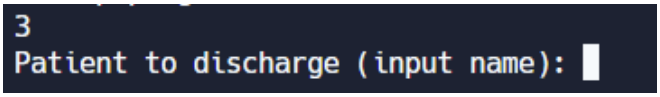
1)



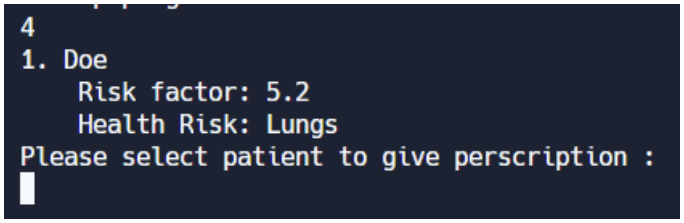

2)



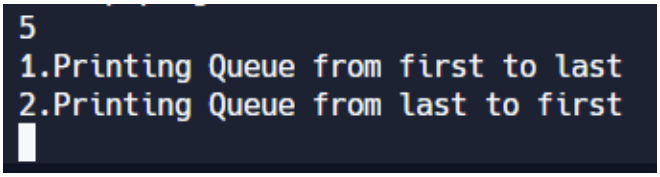

3)



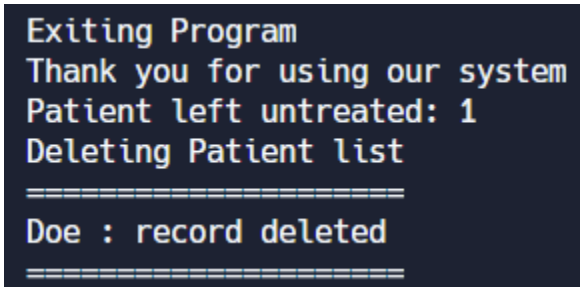

4)




5)




6)



# Work distribution

Akira: Patient Object and patient linked list implementation
Burin: Organ object, organ linked list implementation, basic user interface
Natthaphat : Insertion sort implementation, code optimization
Tanakorn: Information research, calculation implementations and code tester

# Explaining the basics of the program

The data that is required for the calculation of the risk factor are as follows: age, weight (in kilogram), height (in metre), blood pressure (systolic, the top number) and hemoglobin level.

The first data is age. The older you are, the more the risk increases. Unlike younger patients, older patients are expected to experience greater complications and longer recovery periods. The risk that is associated with age is anesthesia. It is a medicine that sedates or causes you to lose consciousness in order to keep you from feeling pain during surgery. The two common risks are postoperative delirium and postoperative cognitive dysfunction. The former is less problematic, causing some confusion, but lasts around a week. The latter, significantly worse, can cause long-term memory loss and make learning, concentrating, and thinking difficult.

The next two data are weight and height, they are used to calculate the BMI. Risks associated with BMI are problem breathing, low oxygen levels, recovery and some surgery. While under anesthesia, you may encounter breathing problems as a result of excess tissue around the neck making it difficult to place a breathing tube. Higher BMI can cause the body to use more oxygen to make the heart pump the blood harder around the body. All of the above will cause the recovery time to be longer than normal. As well as have a massive negative impact on major gastroenterological surgeries.

High blood pressure during surgery has a higher risk for cardiovascular events–events that can cause some damage to the heart–cerebrovascular events–or stroke–bleeding, and death. On the subject of blood, anemia is a disorder in which there are not enough healthy red blood cells to transport enough oxygen to your body's tissues. Which means low hemoglobin level, causing fatigue and other afflictions.

The medicines that are used in the programme are for the heart, the lungs, the kidneys and the liver. There are eight medicines in total, two for each of the four organs.

For the heart are warfarin and azilsartan. Warfarin reduces the blood's capacity to clot (coagulate). Although they are sometimes referred to as blood thinners. Azilsartan prevents a rise in blood pressure by blocking angiotensin II receptors.

For the lungs are theophylline and carbocisteine. Breathing becomes simpler because theophylline's function is to relax and open the lungs' airways. Chest tightness, shortness of breath, and wheezing are all treated and prevented using it. A mucolytic is a class of medication that includes carbocisteine. An aid to coughing out mucus is a mucolytic. It functions by thinning and loosening your mucus.

For the kidneys are azathioprine and prednisolone. For autoimmune diseases like vasculitis, azathioprine is used. It is used to slow the illness and enhance kidney function in an effort to prevent the body from suffering more harm. Prednisolone is used the same as azathioprine, but it is a steroid tablet.

For the liver are ursodiol and propranolol. Patients who do not require the removal of their gallbladders or in whom surgery should be avoided due to other medical issues might utilize ursodiol to dissolve gallstones. In individuals with chronic liver illness, propranolol hydrochloride is said to reduce portal pressure and suppress renin production, activities that may lessen the propensity to ascites development.

# Linked List implementations

## Insertion sort

Different from our sort implementation back during the midterm version of the program, we decided to implement the feature directly in our linked list. When our user chooses inserting mode “-i”, the program would initialize a patient class object and the insertion implementation would find the patient node with less risk factor and insert itself in front of the factor. In this implementation, as we are initializing the patient object and organ object one by one, it means that during each entry of said object, the sort would be done instantly. Linked list allows us to utilize insertion sort ideally and is more reflective of how real world implementation would sort new entries of information.

## Queue

There are two main files we implemented for this part of the program which are; ll.h which is the organ receiving function to see the priority of the patient that needed the organ the most (the most prioritized patient will be deleted from the queue as they already received their much needed organ.) and medrep.cpp that we added one case, the number four that will let the user choose between five choices. The choices include 1. Add patient, 2. Discharge patient, 3. Print current stored patient, 4. Receive organ donation, and 5. Stop the program and delete the record. We start off the mode selection with no information of the patient which the user has to input themselves. For example, inputting the patients various health information and the organ they need and by inputting 3. Print current stored patient will prioritize and print the queue from the first patient to the last. Let’s say there’s a donation that does not match any patient’s requirement, the program will state that there is “No patients waiting for this organ donation”. However, if the requirement meets, the donated organ will be given to the first person on the list and said patient will be deleted from the queue.

The first part of the code that we adjusted is the output part. Instead of 4 choices we made 5 choices instead by having the number four as the receiving organ part and number five as stopping and deleting the record. We added the case 4 in both files which can pass it to ll.h that has the receive organ donating function. In this function, we declared 2 strings which are organ (receive information on what organ is needed) and name (unknown at first because we don't know the patient's name yet). Then, we check if the record is needed or not with the function "is Empty" and print along with return screen and name which is unknown. On the other hand, if it's not empty, we loop from head which has the variable t that will copy link list from the top and let t point to the top of the line; this will loop according to the size of the queue. In addition, the strcmp will compare between the needed organ of the present patient. Note that inorder to use strcmp, we have to transform it into char array first or char* all with string.c_str() in cstring.h. If the organ matches with the needed organ in the record t will get the name of the patient. We let the variable found as 1 at first, if the organ matches the program will memorize the patient's name in "name" and let found be 0 so that the process will not be repeated ( loop with other patients and end with return name). This concludes the patient prioritization search process and return the name. However, if not found the returned name will be unknown. Both type of names will be return to medrep.cpp.

In medrep.cpp that we declared as a string will compare if it's unknown (output will be "No patients waiting for this organ donation") or an actual patient's name (the name will be deleted through the deleted function).

# Double linked list

While the list of patients are either an insertion linked list or a queue linked list (depending on selected mode), it was essential for us to implement double linked list for both mode as we want to allow the functionality of deleting someone and the matched organ from the entire queue with name search or "discharge" function. We go through the patient list and the organ list from first in queue to last, however we are not

deleting them in front to back fashion but rather a “first occurrence”. The first organ donated doesn’t necessarily have to match anyone in the queue, and if that’s the case we should move to the next organ in line as in a surgical sceneriol, we don’t want to be slowing the process down. Same can be said for the patient list, the first patient in the list doesn’t necessarily need the first organ in the list, so we have to look for the patient in the queue that does match the organ. In other words, removing the patient and matching organ could possibly mean removing the objects mid-linked list. Double linked list allow us to remove matching objects easily.

In insertion mode, our linked list of patient is dynamically updated. When the prescription menu is called, the selected patient risk factor would reduce or increase according to prescription. Once risk factor of said patient is updated, the patient’s position in the queue would be reallocated to maintain the descending order of patients’ risk factor. Double linked list allow us to do insertion as well as dynamically changing the patient’s position in the queue.

# Object/Class Implementation

## Patient object

The object (patient) created contains all necessary values needed to identify the patient, calculate the risk factor, and store information.

```
6  class patient{
7    private:
8      patient *nextPatient;
9      patient *prevPatient;
10     string name;
11     int age;
12     string risk;
13     float risk_factor;
14     float weight;
15     float height;
16     float BMI;
17     int blood_pressure;
18     int hemoglobin;
19     int record;
20     string organ;
21
```

```
22   public:
23     patient(string n = "Unnamed", int a = 0, string r =
   "Unknown" , float w  = 0.0, float h = 0.0, int p = 0, int g =
   0, string o = "Unknown");
24     void display();
25     void update_risk_factor();
26     void reduce_risk_factor();
27     float riskfactor();
28     int give_med();
29     void updaterecorded();
30     void updaterecorded(int x){record = x;}
31     patient* getnext();
32     patient* getprev();
33     void setnext(patient*);
34     void setprev(patient*);
35     string getname(){return name;}
36     string getorgan(){return organ;}
37     patient* duplicate();
38     string getrisk(){return risk;}
39     ~patient();
40
```

In the private section, we have all information that is needed for the object that will be created. This information contains name, age, pre-existing condition, weight, height, blood pressure, organ needed from donation, and hemoglobin count. The float risk_factor represents how much health risk a patient has compared to a healthy person. Before the addition of reducing risk, the string risk is not listed to be included into the risk factor calculation, but rather as a note to the doctor. After the addition, string risk will give the doctor an option to give medicine in order to reduce the float risk.

In the public section, constructor and functions used to manipulate the stored value are kept here. In the constructor, we provided standard values to help fill in the blank in case there are unknown values. We have other functions such as give_med, reduce_risk_factor, getorgan, and

getname. Other functions were also included for the addition of the double linked list. We also have a destructor that shows the patient's name before clearing the console screen.

Function display() is for displaying some of the patient values such as name, age, health risk, and health factor. Function update_risk_factor() and riskfactor() work together, with update_risk_factor() calculating the risk factor and riskfactor() updating the calculated float to the database. Function updaterecorded() is used to update the value of int record.

Function give_med() and reduce_risk_factor() works together as well. Function give_med() allows the doctor to give medicine to patients. If the appropriate medicine was given, reduce_risk_factor() function will be used to reduce the risk factor. If wrong medicine was given, reduce_risk_factor() will increase the risk factor. This is to stimulate the real life event where if doctors give the wrong medicine to their patients, it could worsen the patient's conditions.

In the next page, you will see the codes of the functions that were added and modified.

```cpp
int patient::give_med(){
  int i, again = 1;
  cout<<"Please Select the Medicine from the List below: \n";
  cout<<"1. Warfarin\n2. Azilsartan\n3. Theophylline\n4. Carbocisteine\n5. Azathioprine\n6. Prednisolone\n7. Ursodiol\n8. Propranolol\n";
  cin>>i;
  //Warfarin, Azilsartan fur das Herz
  //Theophylline, Carbocisteine fur die Lungen
  //Azathioprine, Prednisolone for the kidneys
  //Ursodiol, Propranolol for liver
  while (again){
    switch(i){
    case 1:
    case 2: again=0; return 1;
      break;
    case 3:
    case 4: again=0; return 2;
      break;
    case 5:
    case 6: again=0; return 3;
      break;
    case 7:
    case 8: again=0; return 4;
      break;
    default: cout<<"No such medicine, please select again."<<endl; again=1;
      break;
    }
  }
}
```

```cpp
void patient::reduce_risk_factor(){
  string r=risk;
  int med;

  transform(r.begin(), r.end(), r.begin(), ::tolower);//I don't think this wor
  med = give_med();
  if(strcmp(r.c_str(),"heart")==0||strcmp(r.c_str(),"Heart")==0){

    if(med==1){
      risk_factor-=5.0;
    } else {
      risk_factor+=10.0;
    }
  } else if(strcmp(r.c_str(),"lungs")==0||strcmp(r.c_str(),"Lungs")==0){

    if(med==2){
      risk_factor-=5.0;
    } else {
      risk_factor+=10.0;
    }
  } else if(strcmp(r.c_str(),"kidneys")==0||strcmp(r.c_str(),"Kidneys")==0){

    if(med==3){
      risk_factor-=5.0;
    } else {
      risk_factor+=10.0;
```

```cpp
patient* patient::duplicate(){
  NodePtr returning = new patient(name, age, risk, weight, height, blood_pressure, hemoglobin, organ);
  record = 5;
  return returning;
}
```

```cpp
    }
  } else if(strcmp(r.c_str(),"liver")==0||strcmp(r.c_str(),"Liver")==0){

    if(med==4){
      risk_factor-=5.0;
    } else {
      risk_factor+=10.0;
    }
  } else {
    cout<<"Medicine not needed, just rest"<<endl;
  }
}
```

```cpp
patient::patient(string n, int a, string r, float w, float h, int p, int g, string o){
  name=n;
  age=a;
  risk=r;
  weight=w;
  height=h;
  blood_pressure=p;
  hemoglobin=g;
  organ=o;
  nextPatient = NULL;
  prevPatient = NULL;
  update_risk_factor();
}
```

# Organ object

The object organ was created to reduce the complexity of the program.

```
class organ{
  private:
    string organname;
    int organquality;
    string donorname;
    organ *nextOrg;
    organ *prevOrg;
    int used;

```

```
  public:
    organ(string,int,string);
    string getorgan(){return organname;}
    int getquality(){return organquality;}
    string getdonorname(){return donorname;}
    void updateused(){used = 1;}
    void setnext(organ*);
    void setprev(organ*);
    organ* getnext(){return nextOrg;}
    organ* getprev(){return prevOrg;}
    ~organ();
};
```

In the private section, we store all the variables we need for registering organs for transplant such as organ name, quality, and donor's name. Int used is implemented to check whether the organ was already used for transplantation. If it was already used, then it will activate the deconstructor.

In the public section, we keep all our functions related to the organ. We have our defaults such as constructor, destructor, setnext, setprev, getnext, and getprev. We also have function gets such as getorgan, getquality, and getdonorname as well due to the values being in the private section. In here there is also updateused function which was introduced in the earlier paragraph. This function's use is incorporated into orgll.h.

```
organ::organ(string in, int quality, string donor){
  organname = in;
  organquality = quality;
  donorname = donor;
  nextOrg = NULL;
  prevOrg = NULL;
  used = 0;
}

```

```
void organ::setnext(organ* t){
   nextOrg = t;
}

void organ::setprev(organ* t){
   prevOrg = t;
}

```

```
organ::~organ(){
  if(!used){
    cout<<"Deleting Organ: "<<organname<<" from system"<<endl;
  }else{
    cout<<"Organ : "<<organname<<"\nDonor name: "
<<donorname<<"\nIs being transplanted"<<endl;
    sleep(5);
    system("clear");
  }
}

```

# Additional Features

The additional features for the updated version of our programme are giving prescriptions, patient organ record and organ donation. The features listed can all be accessed from the main menu.

```
2.Add donor organ
```

```
4.Give perscriptions
```

(This cover both the organ record and donation)

To open the menu for giving prescriptions to patients, type 4 and enter. Then assuming there are already patients in the system, you would pick by typing the number corresponding to the patient.

```
1. Kyle
    Risk factor: 23.27
    Health Risk: Lungs
2. John
    Risk factor: 19.69
    Health Risk: None
Please select patient to give perscription :
```

After picking the patient, you will be presented with eight medicines to prescribe as listed below. These are the medicines for the heart (2 each), the lungs (2 each), the kidneys (2 each) and the liver (2 each).

```
Please Select the Medicine from the List below:
1. Warfarin
2. Azilsartan
3. Theophylline
4. Carbocisteine
5. Azathioprine
6. Prednisolone
7. Ursodiol
8. Propranolol
```

Following the medicine selection, if your patient doesn't have any health risk (string) that the programme recognise, this line will show up:

```
Medicine not needed, just rest
```

However, if the programme recognises the health risk (string), then one of two things will happen. One, if the health risk (string) and the medicine match, then it will reduce the risk factor by 5. On the other hand, if they don't match, it will increase the risk factor by 10.

```
Patient name: Kyle
Age: 87
Additional Health Risk: Lungs
Risk Factor: 18.27
Organ needed: None
======================================
```
(If match)

```
Patient name: Kyle
Age: 87
Additional Health Risk: Lungs
Risk Factor: 33.27
Organ needed: None
======================================
```
(If not match)

For the organ record and organ donation, on the add patient, you should already enter in what organ to transplant. Afterward, go to the second option of the menu to add donors. Then add the donor's organ, name and quality. It will show up on the main menu if the organ does not match any patient's needs. However, if they match, then it will automatically transplant and discharge the patient.

(Read left to right)

```
Organ name: Kidney
Donor name: Doe
Organ quality: 100
```

```
Donor Organ in system:
==========================
Organ : Kidney
Donor : Doe
Quality : 100
==========================
```

```
Patient name: John
Age: 67
Additional Health Risk: None
Risk Factor: 15.07
Organ needed: Liver
==============================

Inserting Queue mode
Donor Organ in system:
==========================
Organ : Kidney
Donor : Doe
Quality : 100
==========================
```

```
Donor organ found
Discharging patient from list : John
Deleting John from list
John : record deleted
Task successful
Returning to main menu
Organ : Liver
Donor name: Doe
Is being transplanted
```

# Summary

By no means is our program an accurate representation of what is used within the medical field. We are still behind in terms of risk coverage, organ coverage and prescription accuracy. By implementing the usefulness of objects and the use of linked lists, we are demonstrating a useful real world use case of linked lists. The flexibility of linked lists means that we don't have to do traditional queueing but still implements the concept of traditional queuing in it. This was possible due the modularity of OOP, optimization of workflows is possible as people come up with abstract ideas for linked lists. We made this program with the "modernization" of the queuing system in traditional hospitals in mind. Picking up the theme of "heal" back in our original midterm submission, we focus on making our program cover more possibilities as well as using linked lists abstractly.

# References

American Heart Association. (n.d.). *Types of Heart Medication.* https://www.heart.org/en/health-topics/heart-attack/treatment-of-a-heart-attack/cardiac-medications

American Society of Anesthesiologists. (n.d.). *Age.* https://www.asahq.org/madeforthismoment/preparing-for-surgery/risks/age/

Aronow W. S. (2017). Management of hypertension in patients undergoing surgery. *Annals of translational medicine*, *5*(10), 227. https://doi.org/10.21037/atm.2017.03.54

Mayo Clinic. (n.d.). *Ursodiol (Oral Route).* https://www.mayoclinic.org/drugs-supplements/ursodiol-oral-route/side-effects/drg-20066618?p=1#:~:text=Ursodiol%20is%20used%20to%20dissolve,because%20of%20other%20medical%20problems.

MedlinePlus. (n.d.). *Theophylline.* https://medlineplus.gov/druginfo/meds/a681006.html#:~:text=Theophylline%20is%20used%20to%20prevent,making%20it%20easier%20to%20breathe.

National Health Service. (n.d.). *Carbocisteine*. https://www.nhs.uk/medicines/carbocisteine/

National Kidney Federation. (n.d.). *COMMONLY PRESCRIBED DRUGS FOR KIDNEY DISEASE.* https://www.kidney.org.uk/commonly-prescribed-drugs-for-kidney-failure#:~:text=to%20fight%20infections.-,All%20patients%20who%20have%20a%20kidney%20transplant%20need%20to%20take,further%20damage%20to%20the%20body.

Oxford University Hospitals NHS Trust. (2015). *Anaesthetic risks if you have a high Body Mass Index (BMI).* https://www.ouh.nhs.uk/patient-guide/leaflets/files/11849Pbmi.pdf

Patel, M. S., & Carson, J. L. (2009). Anemia in the preoperative patient. *The Medical clinics of North America*, *93*(5), 1095–1104. https://doi.org/10.1016/j.mcna.2009.05.007

Rector, W. G., Jr, & Reynolds, T. B. (1984). Propranolol in the treatment of cirrhotic ascites. *Archives of internal medicine*, *144*(9), 1761–1763. https://pubmed.ncbi.nlm.nih.gov/6476994/#:~:text=Propranolol%20hydrochloride%20is%20reported%20to,the%20tendency%20to%20ascites%20formation.

Ri, M., Aikou, S., & Seto, Y. (2017). Obesity as a surgical risk factor. *Annals of gastroenterological surgery*, *2*(1), 13–21. https://doi.org/10.1002/ags3.12049

Whitlock, J. (2021, October 8). *Elderly Patients: Understanding the Risks of Surgery.* verywellhealth. https://www.verywellhealth.com/elderly-patients-and-surgical-risk-4132192#:~:text=They%20are%20at%20higher%20risk,expected%20to%20have%20more%20complications.

Zou, Z., Yuan, H. B., Yang, B., Xu, F., Chen, X. Y., Liu, G. J., & Shi, X. Y. (2016). Perioperative angiotensin-converting enzyme inhibitors or angiotensin II type 1 receptor blockers for preventing mortality and morbidity in adults. *Cochrane Database of Systematic Reviews 2016*, Issue 1. https://doi.org/10.1002/14651858.CD009210.pub2