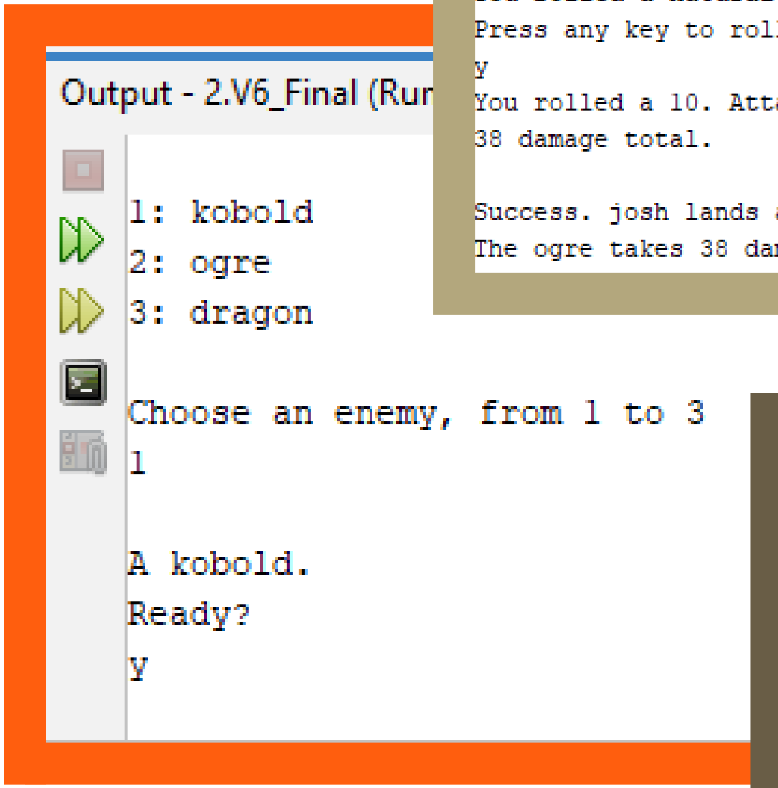
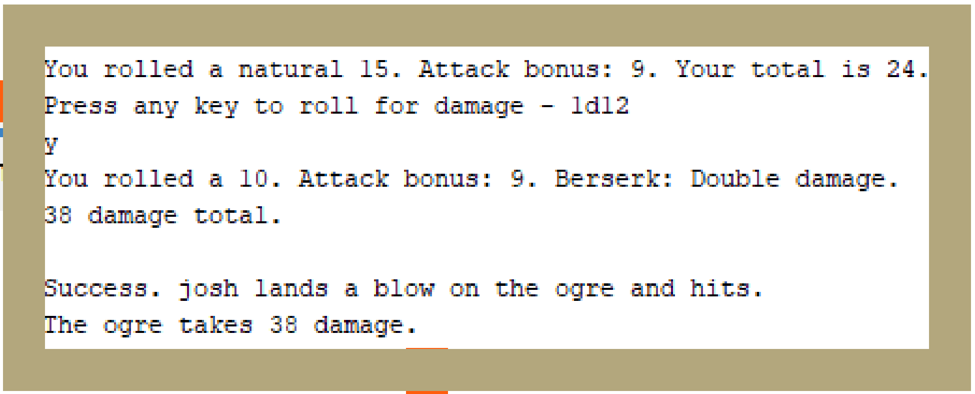
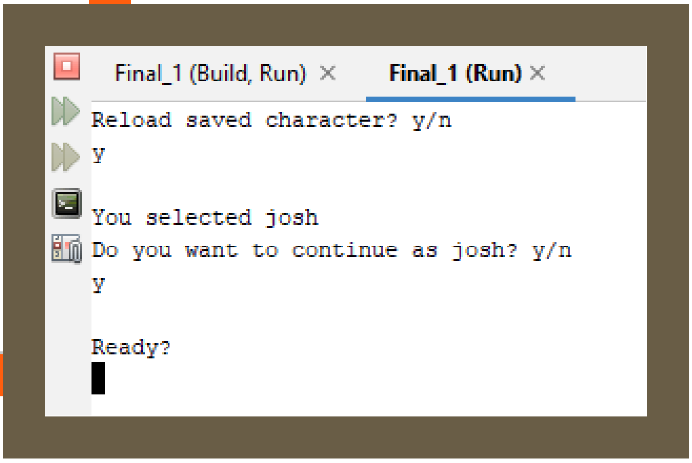
Simplified D&D



```
1: kobold
2: ogre
3: dragon

Choose an enemy, from 1 to 3
1

A kobold.
Ready?
y
```



```
You rolled a natural 15. Attack bonus: 9. Your total is 24.
Press any key to roll for damage - 1d12
y
You rolled a 10. Attack bonus: 9. Berserk: Double damage.
38 damage total.

Success. josh lands a blow on the ogre and hits.
The ogre takes 38 damage.
```



```
Reload saved character? y/n
y

You selected josh
Do you want to continue as josh? y/n
y

Ready?
```

Project 2 Writeup - 7/28/23 - CIS 5 45428

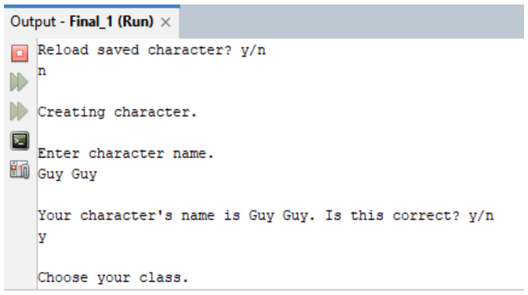
Kaylin Nguyen - Simplified DnD - 750+ lines (~850)

Overview

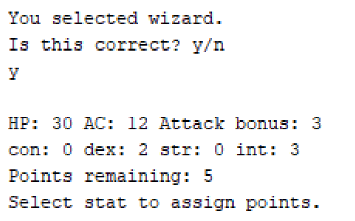
A text based game of a simplified version of Dungeons and Dragons (DnD), meaning it only contains a simple DnD turn based battle mechanic using dice rolls, as well as limited rules, options, and mechanisms. DnD is a tabletop RPG game that utilizes dice rolls to evaluate the success level of a player's action. There are two classes and three enemies to choose from, offering differing options and ranges of difficulty.

Gameplay

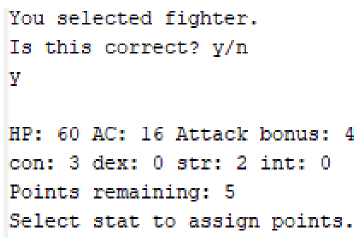
You start by creating your character, by choosing their class and allocating their stats, or by loading one in if you have already created one. When you are done, you enter a battle with an enemy of choice, choosing your actions, which can include attacks or powerups, and rolling for the success of your action, as well as rolling for any damage you output, in a turn based format, until either you or your enemy falls to or below 0 hp.



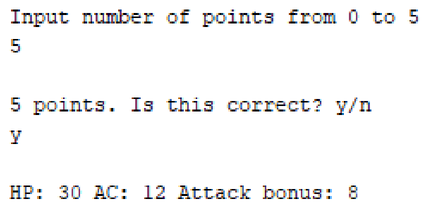
```
Output - Final_1 (Run)
Reload saved character? y/n
n

Creating character.

Enter character name.
Guy Guy

Your character's name is Guy Guy. Is this correct? y/n
y

Choose your class.
```



```
You selected wizard.
Is this correct? y/n
y

HP: 30 AC: 12 Attack bonus: 3
con: 0 dex: 2 str: 0 int: 3
Points remaining: 5
Select stat to assign points.
```

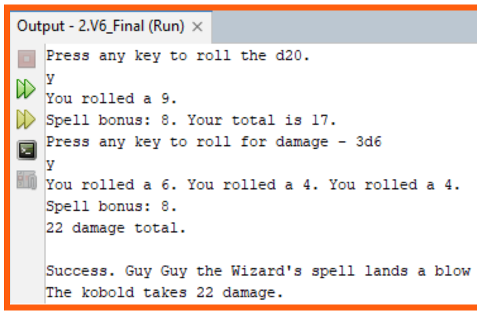


```
You selected fighter.
Is this correct? y/n
y

HP: 60 AC: 16 Attack bonus: 4
con: 3 dex: 0 str: 2 int: 0
Points remaining: 5
Select stat to assign points.
```



```
Input number of points from 0 to 5
5

5 points. Is this correct? y/n
y

HP: 30 AC: 12 Attack bonus: 8
```

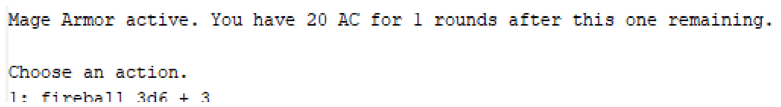
In character creation, the class you choose affects the actions you can take in combat as well as your base stats, and the stats you allocate can further affect your health points (hp), armor class, and attack bonuses which are added to your rolls.



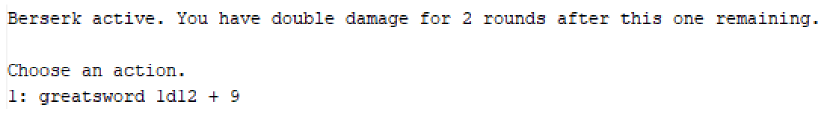
```
Output - 2.V6_Final (Run)
Press any key to roll the d20.
y
You rolled a 9.
Spell bonus: 8. Your total is 17.
Press any key to roll for damage - 3d6
y
You rolled a 6. You rolled a 4. You rolled a 4.
Spell bonus: 8.
22 damage total.

Success. Guy Guy the Wizard's spell lands a blow
The kobold takes 22 damage.
```

Once you choose an enemy, you enter combat. Attack roll totals, including any stat based bonuses, must beat the armor class to hit. If the attack roll is a 20, the hit classifies as a critical hit, and doubles the damage. If the attack roll is 1, it classifies as a critical failure, and the damage is redirected towards the attacker. In either case, you roll for the damage done, and that damage is subtracted from the total hp of the recipient.

Powerups only last for a certain number of rounds, with a counter that decreases each round, and provide a bonus to the player while it is active.

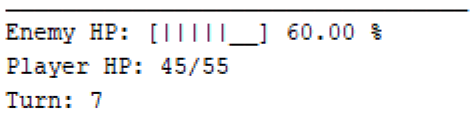


```
Mage Armor active. You have 20 AC for 1 rounds after this one remaining.

Choose an action.
1: fireball 3d6 + 3
```

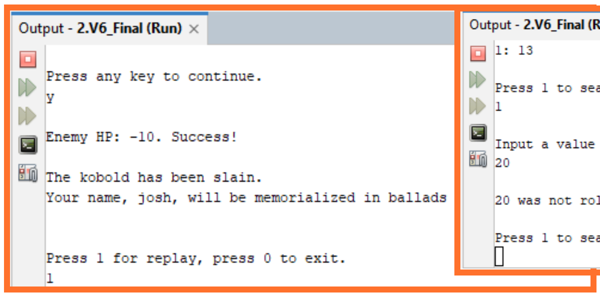


```
Berserk active. You have double damage for 2 rounds after this one remaining.

Choose an action.
1: greatsword 1d12 + 9
```

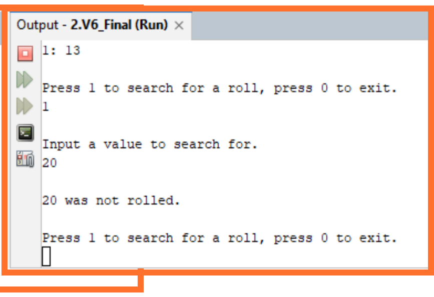
The enemy attacks first in the round, and then the player takes their turn, until either the enemy or the player falls to or below 0 hp. After you are done, you are able to view a replay of your rolls and search for specific rolls.



```
Enemy HP: [|||||__] 60.00 %
Player HP: 45/55
Turn: 7
```



```
Output - 2.V6_Final (Run)

Press any key to continue.
y
Enemy HP: -10. Success!

The kobold has been slain.
Your name, josh, will be memorialized in ballads


Press 1 for replay, press 0 to exit.
1
```



```
Output - 2.V6_Final (Run)
1: 13

Press 1 to search for a roll, press 0 to exit.
1

Input a value to search for.
20

20 was not rolled.

Press 1 to search for a roll, press 0 to exit.
```

Versions

Project 1

Basic gameplay without arrays or functions. A single enemy option, no replays.

V1

Starting part 2 of project, planning how to add functions, arrays, searching and sorting, and writing pseudocode for function to roll dice/damage.

V2

Writing pseudocode for rolls array and determining how top rolls playback would work, testing stub for how to save order, exploring vector.

V3

Adding and adjusting pseudocode based on requirements given, including parallel arrays and two dimensional arrays.

V4

Writing code based on pseudocode for roll function, and sort and search function for vectors. Testing and debugging sorting and binary search using stub.

V5

Writing code for enemies arrays, and sort and print function. Debugging reordering by copying array.

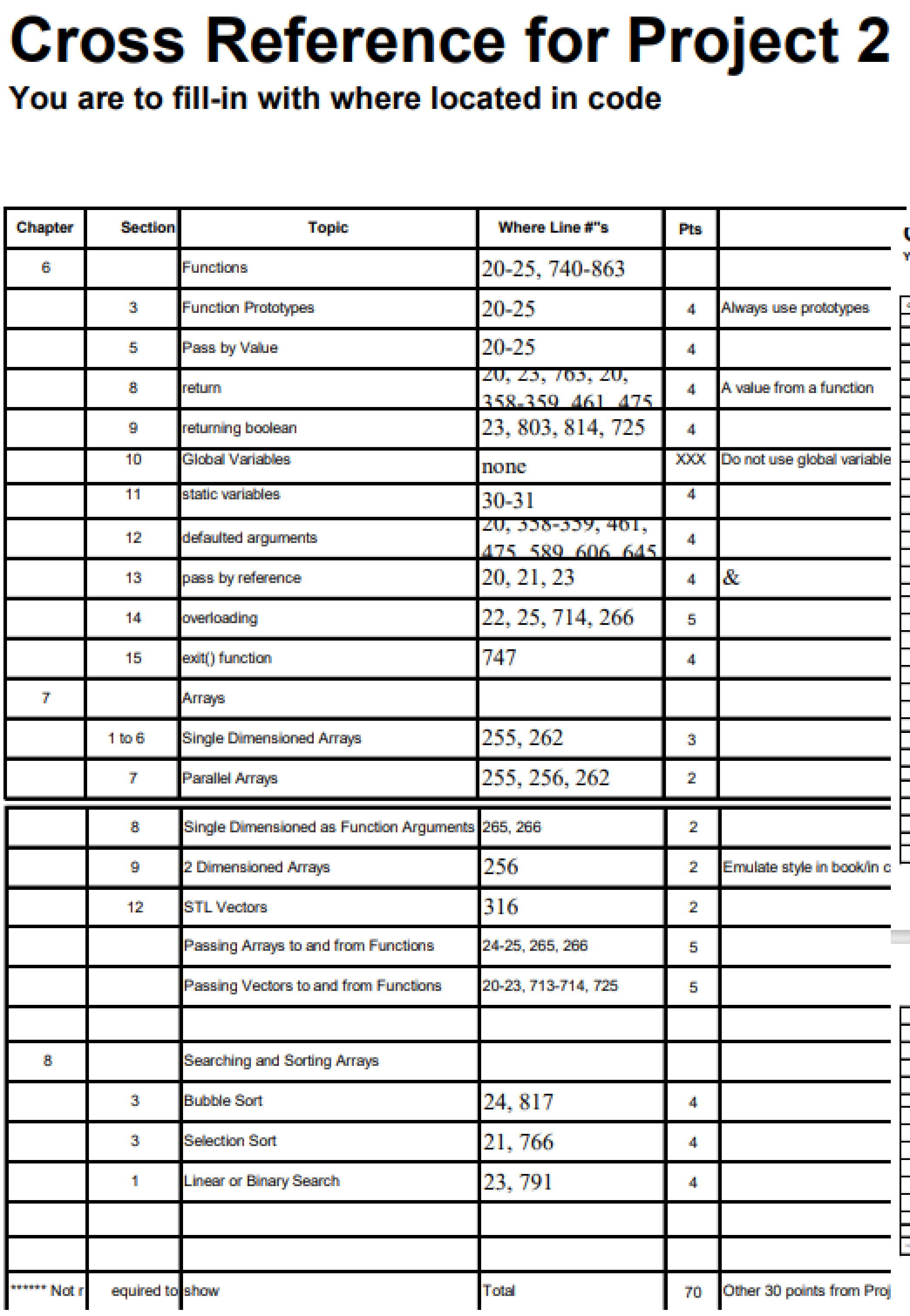
Final Version - V6

Polishing code. Minor adjustments to player experience. Final debugs. Finished product.

Final thoughts

Project seems fully developed. If I had more time, I would have added more diverse dialogue options, and a way to save multiple characters - however such additions go beyond the scope of this project, and the game is quite functional already.

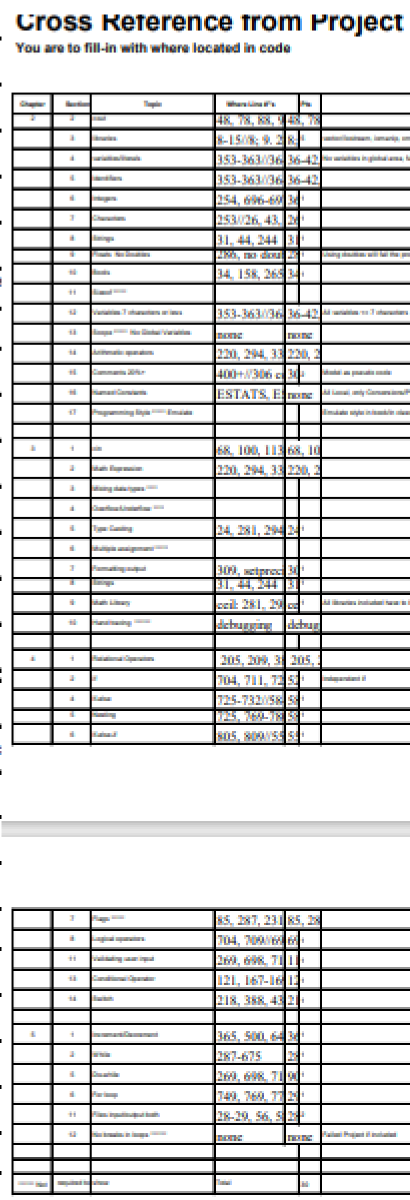
Checkoff Sheet



Cross Reference for Project 2

You are to fill-in with where located in code

Chapter	Section	Topic	Where Line #"s	Pts	
6		Functions	20-25, 740-863		
	3	Function Prototypes	20-25	4	Always use prototypes
	5	Pass by Value	20-25	4	
	8	return	20, 23, 763, 20, 358-359, 461, 475	4	A value from a function
	9	returning boolean	23, 803, 814, 725	4	
	10	Global Variables	none	XXX	Do not use global variable
	11	static variables	30-31	4	
	12	defaulted arguments	20, 358-359, 461, 475, 589, 606, 645	4	
	13	pass by reference	20, 21, 23	4	&
	14	overloading	22, 25, 714, 266	5	
	15	exit() function	747	4	
7		Arrays			
	1 to 6	Single Dimensioned Arrays	255, 262	3	
	7	Parallel Arrays	255, 256, 262	2	
	8	Single Dimensioned as Function Arguments	265, 266	2	
	9	2 Dimensioned Arrays	256	2	Emulate style in book/in c
	12	STL Vectors	316	2	
		Passing Arrays to and from Functions	24-25, 265, 266	5	
		Passing Vectors to and from Functions	20-23, 713-714, 725	5	
8		Searching and Sorting Arrays			
	3	Bubble Sort	24, 817	4	
	3	Selection Sort	21, 766	4	
	1	Linear or Binary Search	23, 791	4	
****** Not r	equired to	show	Total	70	Other 30 points from Proj

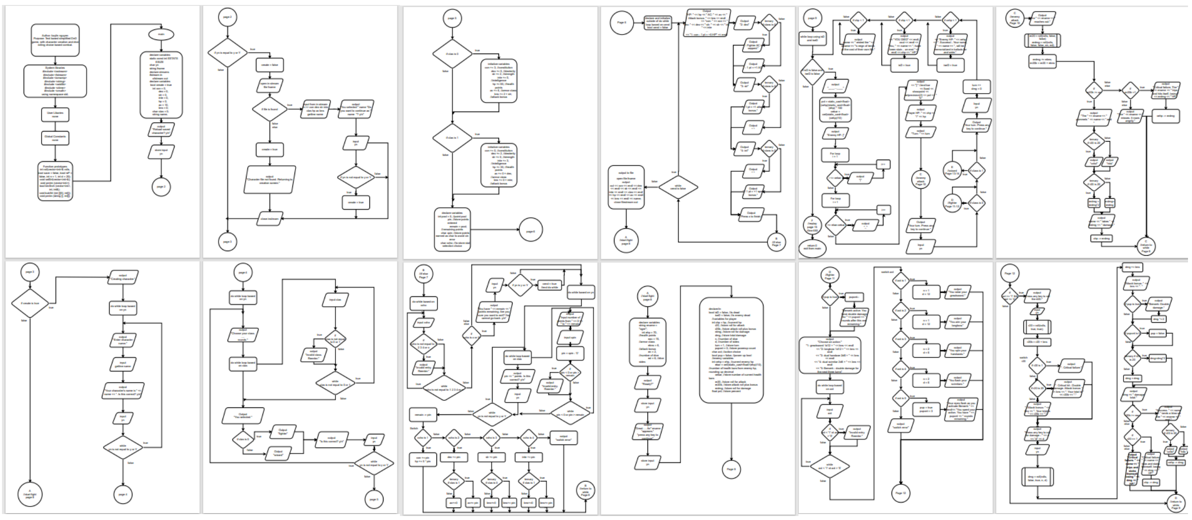


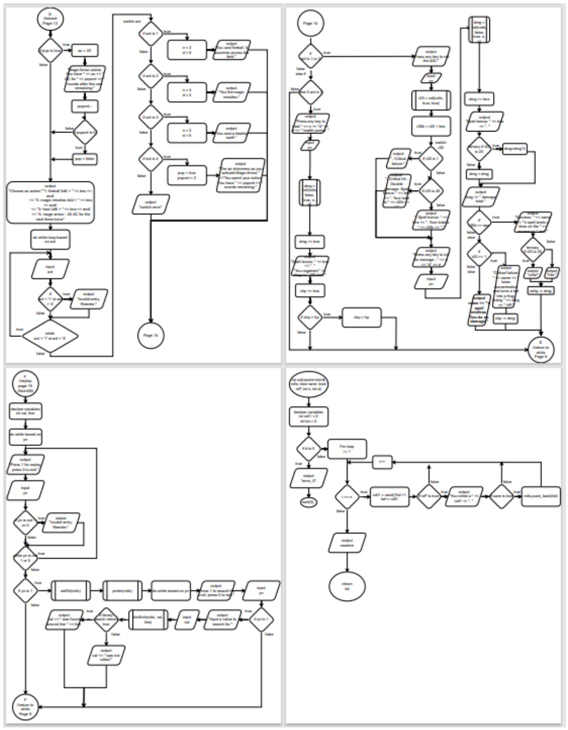
Cross Reference from Project

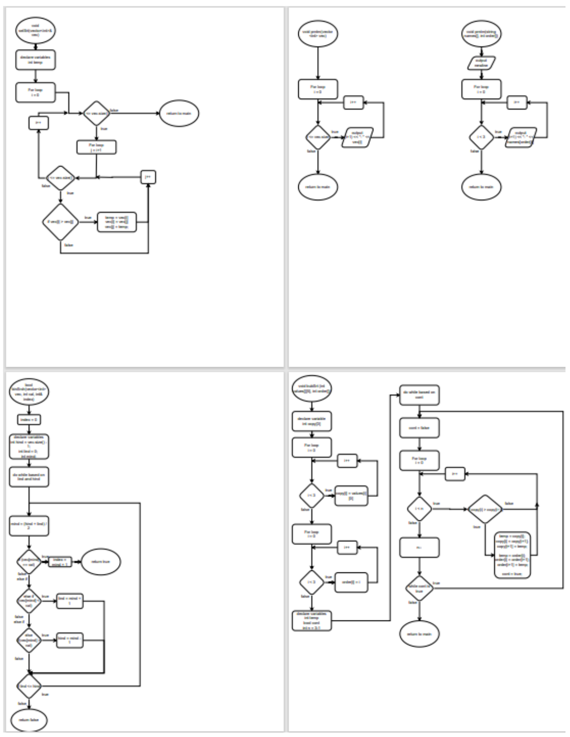
You are to fill-in with where located in code

Chapter	Section	Topic	Where Line #'s	Pts	
2	2	cout	48, 78, 88, 9	48, 78	
	3	libraries	8-15//8; 9, 2	8-	[illegible]
	4	variables/literals	353-363//36	36-42,	[illegible]
	5	Identifiers	353-363//36	36-42,	
	6	Integers	254, 696-69	36	
	7	Characters	253//26, 43,	26	
	8	Strings	31, 44, 244	31	
	9	Floats No Doubles	286, no doub	28	[illegible]
	10	Bools	34, 158, 265	34	
	11	Sizeof ****			
	12	Variables 7 characters or less	353-363//36	36-42,	[illegible]
	13	Scope **** No Global Variables	none	none	
	14	Arithmetic operators	220, 294, 33	220, 2	
	15	Comments 20%+	400+//306 c	30	[illegible]
	16	Named Constants	ESTATS, E	none	[illegible]
	17	Programming Style **** Emulate			[illegible]
3	1	cin	68, 100, 113	68, 10	
	2	Math Expression	220, 294, 33	220, 2	
	3	Mixing data types ****			
	4	Overflow/Underflow ****			
	5	Type Casting	24, 281, 294	24	
	6	Multiple assignment ****			
	7	Formatting output	309, setpreci	30	
	8	Strings	31, 44, 244	31	
	9	Math Library	ceil: 281, 29	ce	[illegible]
	10	Hand tracing ****	debugging	debug	
4	1	Relational Operators	205, 209, 3	205,	
	2	if	704, 711, 72	52	[illegible]
	4	If-else	725-732//58	58	
	5	Nesting	725, 769-78	58	
	6	If-else-if	805, 809//55	55	
	7	Flags ****	85, 287, 231	85, 28	
	8	Logical operators	704, 709//69	69	
	11	Validating user input	269, 698, 71	11	
	13	Conditional Operator	121, 167-16	12	
	14	Switch	218, 388, 43	21	
5	1	Increment/Decrement	365, 500, 64	36	
	2	While	287-675	28	
	5	Do-while	269, 698, 71	90	
	6	For loop	749, 769, 77	25	
	11	Files input/output both	28-29, 56, 5	28	
	12	No breaks in loops ******	none	none	[illegible]
****** Not	required to	show	Total	30	

Flowcharts







Pseudocode

Original Pseudocode - Just the Pieces Added to Project 1* for Project 2

```
//function to roll dice, int roll(vector<int>& rolls, bool save = false, bool isPlayer = false, int
roll number = 1, int sides = 20)
//function to selection sort vector, void void sort (vector<int>& rolls, int size)
//function to print vector void prntm (vector<int>)
//function to search vector, binary search, bool search (vector<int> rolls, int size, int val,
int& index)

//function to bubble sort array (int values[][stats], int size)
//function to print array void prntm (string names[], int size)


/*
    * parallel 1d and 2d arrays for enemies
    * int stats = 5
    * initialize name array = [1, 2, 3]
    * initialize array int values[][stats] size, stats
    * ints for rows cols
    * call sort values by hp
    * call print to print name in new order
    * bubble (values[][stats], size)
    * prntm (names[], size)
    *
    * select enemy
    * do while validate input
    * set stats variables according to input index
*/


/*
    initialize rolls vector
*/


/*
        er20 = roll();
        erdmg += roll(vector, false, false, en, ed)
*/
```

```
/*
            r20 = roll(vector, true, true)
*/


/*
            rdmg = roll(vector, false, true, n, d)
*/


/*
    yn to continue to replay
    call sort function
    call print function
    output press 0 to search for a roll, 1 to exit
    do while yn is not 1
      input yn
      if 0
        input val
        call search function
        if found output found at index
        else output not found
      else output please enter 0 or 1
    if 1
      exit
*/


//int roll(vector<int>& rolls, bool save = false, bool isPlayer = false, int roll number = 1, int
sides = 20)
/*
 * int one roll
 * if sides = 0
 *     output error_0
 *     exit();
 * for loop until n number
 *  one roll = rand()%sides +1
 *  total roll += one roll
 *  if isPlayer is true
 *     output you rolled a roll
 *     if save is true
 *     add to vector roll.push_back(total roll)
 *  return int total roll
 *
*/
//function to selection sort vector, void void sort (vector<int>& rolls, int size)
/*
```

```
 * loop each i, loop i < vector.size()
 * then loop through each after i, so new first to compare to next until first is the smallest
of all
 * if first greater than next
 *  switch first and next
 *  set last to first
*/
//function to print vector
/*
 * for loop through vector.size
    * output
*/
//function to search vector, binary search, bool search (vector<int> rolls, int size, int val,
int& index)
/*
 * var for high, low, middle
    do while dont continue if no changes
        middle index equals high + low /2
        if equal set index to middle, return found true, dont continue
        if (val > mid), new low, continue
        if (val < mid), new high, continue
*/

//function to bubble sort array (int values[][stats], int size)
/*
 * copy array
 * while swapping
    * loop each i until n
    * if first greater than next
       * save first for both copy and order
       * switch first and nest, new first to compare to next until first is the smallest
       * set next to first for both copy and order
       * continue
*/
//function to print array (string names[], int size)
/*
 * use order array
 * for loop through array
    * output based on new order
*/
```

*See Project 1 for remaining original pseudocode.

Code

Project 2 Final Version

```cpp
/*
 * File:   main.cpp
 * Author: knguyen
 * Purpose: Final version of project, testing and polishing code.
 */

//sys lib
#include <iostream>
#include <fstream>   //File I/O
#include <iomanip>   //Format
#include <string>    //String
#include <vector>    //STL Vector Library
#include <cstdlib>   //Rand function
#include <ctime>     //Time to set random function seed
#include <cmath>     //For rounding capability

using namespace std;

//prototypes
int roll(vector<int>& rolls, bool save = false, bool isP = false, int n = 1, int d = 20); //function to roll dice
void selSrt(vector<int>&); //function to selection sort vector
void prntm (vector<int>); //function to print vector
bool binSrch (vector<int>, int, int&); //function to search vector, binary search
void bubSrt (int [][5], int[]); //function to bubble sort
void prntm (string [], int[]); //function to print array

int main(int argc, char** argv)
{
    //constants for array size
    static const int ESTATS = 5,
        ESIZE = 3;
    //Seed rand with the current time
    srand(static_cast<unsigned int>(time(0)));

    char yn; //variable to store yes or no input

    ifstream in; //declare in stream
    ofstream out; //declare out stream
```

```

string fname = "dndchr.dat"; //variable to store file name

//character creation
bool create = true; //bool for creation screen
//variables to store stats
int con = 0, //constitution
        dex = 0, //dexterity
        str = 0, //strength
        inte = 0, //intelligence
        hp = 0, //health points
        ac = 10, //armor class
        bns = 0; //attack bonus
char clas = 0; //class
string name; //name

//load saved character
//prompt to use saved character
cout << "Reload saved character? y/n" << endl;
//store input
cin >> yn;
//conditional to load saved character
if (yn == 'y' || yn == 'Y')
{
    create = false; //toggle create screen
    //open file
    in.open(fname,ios::in);
    //check for file
    if (in.fail() == false) //if file can be found
    {
        //store input from file
        in >> con >> dex >> str >> inte >> clas >> hp >> ac >> bns; //int values
        in.ignore(); //ignore endl
        getline(in, name); //getline for string
        //verify choice
        cout << endl << "You selected " << name << endl
                << "Do you want to continue as " << name << "? y/n" << endl;
        //store input
        cin >> yn;
        if (yn != 'y' && yn != 'Y') //if not yes
        {
            create = true; //toggle character creation
        }
    }
    else //if file cannot be found
    {
        create = true; //toggle character creation
```

```
            //inform player
            cout << endl << "Character file not found. Returning to creation screen." << endl;
        }
        //close file
        in.close();
    }

    //character creation screen
    if (create == true)
    {
        //inform player
        cout << endl << "Creating character." << endl << endl;
        //character name
        do
          {
            //prompt input
            cout << "Enter character name." << endl;
            //store character name
            cin.ignore(); //ignore last endl
            getline(cin, name); //store name from cin
            //confirm name
            cout << endl << "Your character's name is " << name << ". Is this correct? y/n" << endl;
            //store input
            cin >> yn;
          } while (yn != 'y' && yn != 'Y'); //confirm entry with do while

          //choose class
          do
          {
            //prompt input
            cout << endl << "Choose your class." << endl << "Select 0 for Fighter and 1 for Wizard." << endl
                    << "0: Fighter. 60 HP. +2 con +3 str, +2 attack bonus. Armor - 16 AC, limits dex. Berserk - double damage, 3 rounds." << endl
                    << "1: Wizard.  30 HP. +3 int +2 dex. Spells include Healing, and Mage Armor - 20 AC, 3 rounds." << endl;
            do
            {
                //store class
                cin >> clas;
                //output invalid message
                if (clas != '0' && clas != '1')
                {
                    cout << endl << "Invalid class. Reenter." << endl;
                }
            } while (clas != '0' && clas != '1'); //validate entry with do while
            //confirm entry
            cout << "You selected " << ( (clas == '0')? "fighter" : "wizard" ) << "." << endl
                    << "Is this correct? y/n" << endl;
            //store input
            cin >> yn;
          } while (yn != 'y' && yn != 'Y'); //confirm entry with do while

          //set base stats based on class chosen
          //set fighter stats
          if (clas == '0')
          {
            con += 3, //constitution
            dex += 0, //dexterity
            str += 2, //strength
            inte += 0, //intelligence
            hp += 60, //health points
            ac += 6, //armor class
            bns += 2 + str; //attack bonus
          }
          //set wizard stats
          if (clas == '1')
          {
            con += 0, //constitution
            dex += 2, //dexterity
            str += 0, //strength
            inte += 3, //intelligence
            hp += 30, //health points
            ac += 0 + dex, //armor class
            bns += 0 + inte; //attack bonus
          }

          //choose stats
          //variable to store point pool, stat choice, point allocation input, end bool
          int pool = 5, //point pool
                pin, //store points entered
                remain = pool; //remaining points
          char cpin; //store points earned as char to avoid cin error
          char ccho; //to store stat selection choice
          bool cend = false; //to end stat allocation
          do //do until exit confirmed
          {
            //prompt input to assign points, displaying info
            cout << endl << "HP: " << hp << " AC: " << ac << " Attack bonus: " << bns << endl
                    << "con: " << con << " dex: " << dex << " str: " << str << " int: " << inte << endl
                    << "Points remaining: " << remain << endl
                    << "Select stat to assign points." << endl
                    << "1: con - 1 pt = +5 HP" << endl
                    << "2: dex" << ( (clas == '0')? " - Fighter AC is capped." : " - 1 pt = +1 AC" ) << endl
                    << "3: str" << ( (clas == '0')? " - 1 pt = +1 attack bonus." : "" ) << endl
                    << "4: int" << ( (clas == '0')? "" : " - 1 pt = +1 spell bonus." ) << endl
                    << "Press x to finish." << endl;
            do
            {
                //store input char choice
                cin >> ccho;
                //output invalid message
                if (ccho != '1' && ccho != '2' && ccho != '3' && ccho != '4' && ccho != 'x' && ccho != 'X')
                {
                    cout << endl << "Invalid entry. Reenter." << endl;
                }
            } while (ccho != '1' && ccho != '2' && ccho != '3' && ccho != '4' && ccho != 'x' && ccho != 'X'); //validate with do while

            if (ccho == 'x' || ccho == 'X') //if exit chosen
            {
                //confirm exit
                cout << endl << "You have " << remain << " points remaining. Are you sure you want to exit? You cannot go back. y/n" << endl;
                //store input
                cin >> yn;
                if (yn == 'y' || yn == 'Y') //if yes
                {
                    cend = true; //toggle exit, stat screen off
                }
            }
            else
            {
                do //confirm entry
                {
                    do //validate input
                    {
                        //prompt input
                        cout << endl << "Input number of points from " << 0 << " to " << remain << endl;
                        //store char input
                        cin >> cpin;
                        pin = cpin - '0'; //clean and convert char to int
                        //output invalid message if pin not in range
                        if ( (pin < 0 || pin > remain) )
```

```
                    {
                        cout << endl << "Invalid entry. Reenter." << endl;
                    }
                } while ( pin < 0 || pin > remain ); //validate input with do while
                //confirm entry
                cout << endl << pin << " points. Is this correct? y/n" << endl;
                //store input
                cin >> yn;
            } while (yn != 'y' && yn != 'Y'); //confirm entry with do while
            //update remaining points
            remain -= pin;
            //switch to update stats
            switch (ccho)
            {
                case '1': con += pin; hp += 5 * pin; //update con and hp
                    break;
                case '2': dex += pin; ac += ( (clas == '1')? pin : 0 ); //update dex and ac if wizard
                    break;
                case '3': str += pin; bns += ( (clas == '0')? pin : 0 ); //update str and bonus if fighter
                    break;
                case '4': inte += pin; bns += ( (clas == '1')? pin : 0 ); //update str and bonus if wizard
                    break;
                default: cout << endl << "switch error" << endl;
            }
        }
    } while (cend == false); //do while stat screen is toggled on

    //store character information in file
    //open file
    out.open(fname,ios::out);
    //output into file
    out << con << endl << dex << endl << str << endl << inte << endl << clas << endl << hp << endl << ac << endl << bns << endl << name;
    //close file
    out.close();
}

//enemy stats
//variables to store enemy stats
char cenemy; //store enemy choice as char
int enemy; //enemy choice as int
string eaname[ESIZE] = {"ogre", "dragon", "kobold"}; //enemy names
int evals[ESIZE][ESTATS] = //enemy stats
{
    {70, 15, 6, 2, 6},
    {130, 17, 8, 3, 8},
    {10, 14, 4, 1, 8}
};
int order[ESIZE]; //array to store new order
int eind; //set order input
//sort and print
bubSrt(evals, order); //sort in order
prntm(eaname, order); //print in order, starting at 1

//choose enemy
do
{
    cout << endl << "Choose an enemy, from 1 to " << ESIZE << endl; //prompt input
    cin >> cenemy; //store input
    enemy = cenemy - '0'; //convert input to integer
    if (enemy < 1 && enemy > ESIZE) //if invalid
    {
        cout << endl << "Invalid entry. Reenter." << endl; //output invalid message
    }
} while (enemy < 1 && enemy > ESIZE); //do while validate

//set enemy stats
eind = order[enemy - 1]; //index equals enemy choice - 1, in spot saved by new 
order array
string ename = eaname[eind]; //enemy name
int ehp = evals[eind][0], //health points
        eac = evals[eind][1], //armor class
        ebns = evals[eind][2], //attack bonus
        en = evals[eind][3], //number of dice
        ed = evals[eind][4]; //dice sides
//output beginning of fight
cout << endl << "A " << ename << "." << endl << "Ready?" << endl;
cin >> yn;
cout << endl << "Great." << endl
        << "..." << endl
        << "A " << ename << " appears.";

//prompt continue
cout << endl << endl << "Press any key to continue." << endl;
//enter to continue
cin >> yn;

//fight mechanics
//variables to keep track of battle
bool isD = false, //is dead
    iseD = false; //is enemy dead
```

```
//variables for player
int chp = hp, //current hp
   r20, //store roll for attack
   r20b, //store attack roll plus bonus
   rdmg, //store roll for damage
   dmg, //store total damage
   n, //number of dice
   d, //number of sides
   turn = 1, //store turn
   pupcnt = 0; //store powerup count
char act; //action choice
bool pup = false; //power up bool
vector<int> rolls;
//enemy variables
int cehp = ehp, //current enemy hp
   ebar = ceil(static_cast<float>(ehp)/10), //number of health bars from enemy hp, rounding up decimal
   cebar, //store number of current health bars
   er20, //store roll for attack
   er20b, //store attack roll plus bonus
   erdmg; //store roll for damage
float pct; //store percent
while (isD == false && iseD == false) //while both are alive
{
   //Mark new turn
   cout << "____________________________________";

   //display enemy health percentage bar
   //store calculations
   pct = static_cast<float>(cehp)/static_cast<float>(ehp) * 100.0f; // health percentage
   cebar = ceil(static_cast<float>(cehp)/10); //current bars, rounding up decimal
   //start output
   cout << endl << "Enemy HP: [";
   //loop through current health bars
   for (int i = 1; i <= cebar; i++)
   {
      cout << "|";
   }
   //loop through depleted health bars
   for (int i = 1; i <= ( ebar - cebar ); i++)
   {
      cout << "_";
   }
   cout << "] " //end bar
         << fixed << showpoint << setprecision(2) << pct << " %" << endl; //output percent at 2 points
```

```

    //display player current / hp
    cout << "Player HP: " << chp << "/" << hp << endl;

    //display turn
    cout << "Turn: " << turn << endl;

    //enemy attack
    //output attack
    cout << endl << "The " << ename << " lunges at you." << endl;
    er20 = roll(rolls, false, false); //roll for attack
    erdmg = roll(rolls, false, false, en, ed); //roll for damage
    erdmg += ebns; //roll for damage + bonus
    er20b = er20 + ebns; //attack roll plus bonus
    if (er20b >= ac) //if roll > ac, hits
    {
        //output attack line, if natural 20 critical hit
        cout << endl << "The " << ename << " pummels " << name << " and " << ( (er20 == 20)? "crits! " : "hits. ");
        //double damage if critical
        erdmg = (er20 == 20)? erdmg*2 : erdmg;
        //output damage done
        cout << name << " takes " << erdmg << " damage." << endl;
        //update player hp
        chp -= erdmg;
    }
    else
    {
        if (er20 == 1) //if natural roll is 1, critical failure
        {
            //output critical failure
            cout << endl << "Critical failure. The " << ename << " trips and hits itself, losing " << erdmg << " HP." << endl;
            //update enemy hp
            cehp -= erdmg;
        }
        else //else normal miss
        {
            //output dialogue
            cout << endl << "The " << ename << " misses. It roars angrily." << endl;
        }
    }

    //player attack
    //begin output
    cout << endl << "Your turn. Press any key to continue." << endl;
    //enter to continue
```

```
cin >> yn;
//action switch
//if fighter
if (clas == '0')
{
   if (pup == true) //if powerup is active
   {
      pupcnt--; //decrease count per round
      cout << endl << "Berserk active. You have double damage for " << pupcnt << 
" rounds after this one remaining." << endl; //inform player
   }
   //prompt action choice
   cout << endl << "Choose an action." << endl;
   //display menu
   cout << "1: greatsword 1d12 + " << bns << endl
         << "2: longbow 1d12 + " << bns << endl
         << "3: dual handaxe 2d6 + " << bns << endl
         << "4: dual scimitar 2d6 + " << bns << endl
         << "5: Berserk - double damage for the next three turns" << endl;
   //store choice
   do
   {
      //store action
      cin >> act;
      if (act < '1' || act > '5')
      {
         //invalid message
         cout << endl << "Invalid entry. Reenter." << endl;
      }
   } while (act < '1' || act > '5'); //validate input
   //switch choice
   switch (act)
   {
      //weapons, number of dice and sides
      case '1':
         n = 1;
         d = 12;
         cout << endl << "You raise your greatsword." << endl;
         break;
      case '2':
         n = 1;
         d = 12;
         cout << endl << "You aim your longbow." << endl;
         break;
      case '3':
         n = 2;
         d = 6;
```

```
                cout << endl << "You spin your handaxes." << endl;
                break;
            case '4':
                n = 2;
                d = 6;
                cout << endl << "You flash your scimitars." << endl;
                break;
            case '5': //berserk
                pup = true; //toggle powerup on
                pupcnt = 3; //set count for rounds remaining
                cout << endl << "Your eyes flash as you activate Berserk." << endl
                        << "You spent your action. You have " << pupcnt << " rounds
remaining.";
                break;
            default: cout << "switch error";
        }
        if (act >= '1' && act <= '4') //if attack action
        {
            //roll to attack
            //prompt roll
            cout << endl << "Press any key to roll the d20." << endl;
            //enter for "roll"
            cin >> yn;
            r20 = roll(rolls, true, true); //roll d20, save roll, output
            r20b = r20 + bns; //attack roll plus bonus
            //switch output depending on natural roll
            switch (r20)
            {
                case 1: cout << "Critical failure."; break;
                case 20: cout << "Critical hit - Double damage. Attack bonus: " << bns << ".
Your total is " << r20b << "."; break;
                default: cout << "Attack bonus: " << bns << ". Your total is " << r20b << ".";
            }

            //prompt damage roll
            cout << endl << "Press any key to roll for damage - " << n << "d" << d <<
endl;
            //enter for "roll"
            cin >> yn;
            dmg = roll(rolls, false, true, n, d); //roll for damage
            dmg += bns; //total roll for damage + bonus
            cout << "Attack bonus: " << bns << ". "; //output bonus
            if (pup == true) //if berserk is on
            {
                cout << "Berserk: Double damage."; //output berserk notification
                dmg *= 2; //double damage
            }
            //if count reaches zero
            if (pupcnt == 0)
            {
                pup = false; //toggle berserk off
            }
            //double damage if critical
            dmg = (r20 == 20)? dmg*2 : dmg;
            //output total damage
            cout << endl << dmg << " damage total." << endl;

            if (r20b >= eac) //if roll meets or exceeds ac, hits
            {
                //output attack line, if natural 20 critical hit
                cout << endl << "Success. " << name << " lands a blow on the " << ename << " and " << ( (r20 == 20)? "crits!" : "hits.") << endl;
                //output damage done
                cout << "The " << ename << " takes " << dmg << " damage." << endl;
                //update player hp
                cehp -= dmg;
            }
            else
            {
                if (r20 == 1) //if natural roll is 1, critical failure
                {
                    //output critical failure
                    cout << endl << "Critical failure. " << name << " trips and stabs themself, losing " << dmg << " HP." << endl;
                    //update enemy hp
                    chp -= dmg;
                }
                else //else normal miss
                {
                    //output dialogue
                    cout << endl << name << " leaps forward and misses. You do no damage." << endl;
                }
            }
        }
    }

    //if wizard
    if (clas == '1')
    {
        if (pup == true) //if powerup is active
        {
            ac = 20; //set armor class to 20
            cout << endl << "Mage Armor active. You have " << ac << " AC for " <<
```

```
pupcnt << " rounds after this one remaining." << endl; //inform player
            pupcnt--; //decrease count per round
            if (pupcnt == 0) //if count reaches zero
            {
                pup = false; //toggle armor off
            }
        }

        //prompt action choice
        cout << endl << "Choose an action." << endl;
        //display menu
        cout << "1: fireball 3d6 + " << bns << endl
            << "2: magic missiles 4d4 + " << bns << endl
            << "3: heal 2d6 + " << bns << endl
            << "4: mage armor - 20 AC for the next three turns" << endl;
        //store choice
        do
        {
            //store action
            cin >> act;
            //output invalid message
            if (act < '1' || act > '4')
            {
                cout << endl << "Invalid entry. Reenter." << endl;
            }
        } while (act < '1' || act > '4'); //validate input
        //switch choice
        switch (act)
        {
            //weapons, number of dice and sides
            case '1':
                n = 3;
                d = 6;
                cout << endl << "You cast fireball. It launches across the field." << endl;
                break;
            case '2':
                n = 4;
                d = 4;
                cout << endl << "You fire magic missiles." << endl;
                break;
            case '3':
                n = 2;
                d = 6;
                cout << endl << "You cast a healing spell." << endl;
                break;
            case '4': //mage armor
                pup = true; //toggle powerup on
                pupcnt = 3; //set count for rounds remaining
                cout << endl << "The air shimmers as you activate Mage Armor." << endl
                        << "You spent your action. You have " << pupcnt << " rounds remaining.";
                break;
            default: cout << "switch error";
        }

        //execute actions with rolls
        if (act == '1' || act == '2')
        {
            //roll to attack
            //prompt roll
            cout << endl << "Press any key to roll the d20." << endl;
            //enter for "roll"
            cin >> yn;
            //roll for attack
            r20 = roll(rolls, true, true); //roll d20, save roll, output
            r20b = r20 + bns; //attack roll plus bonus
            //switch output depending on natural roll
            switch (r20)
            {
                case 1: cout << "Critical failure."; break;
                case 20: cout << "Critical hit - Double damage. Spell bonus: " << bns << ". Your total is " << r20b << "."; break;
                default: cout << "Spell bonus: " << bns << ". Your total is " << r20b << ".";
            }

            //prompt damage roll
            cout << endl << "Press any key to roll for damage - " << n << "d" << d << endl;
            //enter for "roll"
            cin >> yn;
            //roll for attack
            dmg = roll(rolls, false, true, n, d); //roll for damage
                //also outputs roll

            dmg += bns; //total roll for damage + bonus
            cout << "Spell bonus: " << bns << ". "; //output bonus
            //double damage if critical
            dmg = (r20 == 20)? dmg*2 : dmg;
            //output total damage
            cout << endl << dmg << " damage total." << endl;

            if (r20b >= eac) //if roll meets or exceeds ac, hits
            {
                //output attack line, if natural 20 critical hit
```

```
                cout << endl << "Success. " << name << "'s spell lands a blow on the " << ename << " and " << ( (r20 == 20)? "crits!" : "hits.") << endl;
                //output damage done
                cout << "The " << ename << " takes " << dmg << " damage." << endl;
                //update player hp
                cehp -= dmg;
            }
            else
            {
                if (r20 == 1) //if natural roll is 1, critical failure
                {
                    //output critical failure
                    cout << endl << "Critical failure. " << name << " loses concentration and turns a toe into a frog, losing " << dmg << " HP." << endl;
                    //update enemy hp
                    chp -= dmg;
                }
                else //else normal miss
                {
                    //output dialogue
                    cout << endl << name << "'s spell misfires. You do no damage." << endl;
                }
            }
        }
        else if (act == '3') //else if 3 you cast heal, roll to heal, adjust hp
        {
            //prompt healing roll
            cout << endl << "Press any key to heal " << n << "d" << d << " health points. " << endl;
            //enter for "roll"
            cin >> yn;
            dmg = roll(rolls, false, true, n, d); //roll for healing points
            dmg += bns; //total roll for healing + bonus
            cout << "Spell bonus: " << bns << ". "; //output bonus
            //output total healing points
            cout << endl << "You regained " << dmg << " HP total." << endl;
            //update hp
            chp += dmg;
            if (chp > hp) //if healing points lead to current hp exceeding max hp
            {
                chp = hp; //set equal to max hp
            }
        }
    }

    //prompt continue
    cout << endl << "Press any key to continue." << endl;
```

```
        //enter to continue
        cin >> yn;

        //update turn
        turn ++;
        //reset damage
        dmg = 0;

        //if enemy dead
        if (cehp < 1)
        {
            //output results
            cout << endl << "Enemy HP: " << cehp << ". Success!" << endl
                    << endl << "The " << ename << " has been slain." << endl
                    << "Your name, " << name << ", will be memorialized in ballads far and wide." << endl;
            iseD = true; //toggle death end screen
        }
        //if you are dead
        if (chp < 1)
        {
            //output results
            cout << endl << "YOU DIED" << endl;
            cout << endl << "You, " << name << ", have been slain. Your final act was injuring the " << ename << "." << endl
                    << "However, your journey has come to an end." << endl << chp << " HP." << endl;
            isD = true; //toggle death end screen
        }
        //if both dead
        if (chp < 1 && cehp < 1)
        {
            //output results
            cout << name << " ended the " << ename << "'s reign of terror, at the cost of their own life." << endl;
        }
    }

    //replay screen
    int val, //to store value
            line; //to store line found
    do
    {
        //prompt input
        cout << endl << endl << "Press 1 for replay, press 0 to exit." << endl;
        //store input
        cin >> yn;
        if (yn != '1' && yn != '0') //if invalid
        {
            //output invalid message
            cout << "Invalid entry. Reenter.";
        }
    } while (yn != '1' && yn != '0');

    if (yn == '1') //if replay
    {
        selSrt(rolls); //selection sort rolls
        prntm(rolls); //print rolls in order from least to greatest
        do
        {
            //prompt input
            cout << endl << "Press 1 to search for a roll, press 0 to exit." << endl;
            //store input
            cin >> yn;
            if (yn == '1') //if searching
            {
                cout << endl << "Input a value to search for." << endl; //prompt input
                cin >> val; //store input
                if( binSrch(rolls, val, line) == true ) //if found
                {
                   cout << endl << val << " was found around line " << line << endl; //output result
                }
                else //if not found
                {
                   cout << endl << val << " was not rolled." << endl; //output result
                }
            }
        } while (yn == '1'); //while searching
    }
    //exit
    return 0;
}

int roll(vector<int>& rolls, bool save, bool isP, int n, int d)
{
    int roll1 = 0; //single roll
    int tot = 0; //total roll
    if (d == 0) //if sides is 0
    {
        cout << endl << "error_0" << endl; //output error, unable to divide by 0
        exit(0); //exit
    }
    for (int i = 1; i <= n; i++) //roll for damage for loop number of dice
    {
        roll1 = rand()%d +1; //damage roll
        tot += roll1; //add to total damage
        if (isP == true) //if player
        {
            cout << "You rolled a " << roll1 << ". "; //output roll
            if (save == true) //if saving is on
            {
                rolls.push_back(tot); //add to vector
            }
        }
    }
    cout << endl; //final endl
    return tot; //return total roll
}

void selSrt(vector<int>& vec)
{
    int temp;
    for (int i = 0; i < vec.size(); i++) //loop each i until end of vector
    {
        for (int j = i+1; j < vec.size(); j++) //loop through each after i
        {
            if (vec[i] > vec[j]) //if greater than next
            {
                temp = vec[i]; //save first
                vec[i] = vec[j]; //switch first and last, first to compare to next until first is the smallest
                vec[j] = temp; //set last to first
            }
        }
    }
}

void prntm(vector <int> vec)
{
    for (int i = 0; i < vec.size(); i++) //loop through vector
    {
        cout << (i+1) << ": " << vec[i] << endl; //output list number and roll
    }
}

bool binSrch(vector<int> vec, int val, int& index)
{
    index = 0; //set index
    int hind = vec.size() - 1; //high
    int lind = 0; //low
    int mind; //mid
    do
    {
       mind = (hind + lind) / 2; //mid
       if (vec[mind] == val) //if value equals middle value
       {
          index = mind + 1; //save index
          return true; //if found true
       }
       else if (vec[mind] < val) //if less
       {
          lind = mind + 1; //mid is new high
       }
       else if (vec[mind] > val) //if greater
       {
          hind = mind - 1; //mid is new low
       }
    } while (lind <= hind); //until search finished
    return false; //return bool if found
}

void bubSrt (int values[][5], int order[])
{
    int copy[3]; //array to copy
    //copy array
    for (int i = 0; i < 3; i++)
    {
       copy[i] = values[i][0]; //set to each first num in value array
    }
    //initialize order array
    for (int i = 0; i < 3; i++)
    {
       order[i] = i;
    }

    int temp; //store temporary
    bool cont; //bool to continue
    int n = 3-1; //n to loop to
    do
    {
       cont = false; //dont continue if no swaps
       for (int i = 0; i < n; i++) //loop each i
       {
          if (copy[i] > copy[i+1]) //if greater than next
          {
             //sort copy
             temp = copy[i]; //save first
```

```
                copy[i] = copy[i+1]; //switch first and nest, new a[i] to compare to next j until ai is the smallest
                copy[i+1] = temp; //set next to first
                //update order
                temp = order[i]; //save first
                order[i] = order[i+1]; //switch first and nest, new a[i] to compare to next j until ai is the smallest
                order[i+1] = temp; //set next to first
                cont = true; //continue
            }
        }
        n--; //last is largest
    } while (cont == true); //while still swapping
}

void prntm (string names[], int order[])
{
    cout << endl << endl; //new line
    for (int i = 0; i < 3; i++) //loop through array in new order
    {
        cout << (i+1) << ": " << names[order[i]] << endl; //output array
    }
}
```

Conclusion

This project for text based DnD relies on input and output, switching cases between choices, as well as do while loops to verify inputs, if and else statements including nested statements to regulate mechanisms. In the peripherals are file input and output, for loops for repetitive tasks, varying variable types, varying operations and operators, and arrays.

Adding functions streamlined the process of calculating and outputting rolls for various boolean instances. Arrays and vectors aided in saving data to later output and sort and search, enhancing overall gameplay and enabling expansion of options.