Introduction:

Title: Treasure Glacier!

Treasure Glacier is mathematics based role-playing-game (RPG) where the player has to solve math problems and questions to advance in the game. When the user starts, they are given a brief preposition and a set of instructions which they have the option of skipping if they have played before. The player starts out on the first of three levels, they have the option of catching an elf with a scroll, catching a normal elf, going to the unique shop (think of shop in terms of an autoshop or woodshop) for that level, going to the store, looking for treasures, and finally, ascending to the next level. All the floors are the same except for the shop. The first floor shop asks questions based on clock logic, the player is asked what the difference is between two times, and finds the difference in hours and minutes. The Second shop is the Scale shop which is based on simple algebra, the player is asked to guess the mass of a rock between 1 and 30 kg, by inputting a number. The player is then told if their guess is greater or less than the rock, any inputs afterward will add or subtract from their previous guess, as opposed to simply inputting a lower or greater value. For example if a player is told that the rock is less than their guess of 15kgs, and they want to guess 14kg, the player inputs -1, not 14.

The level has the Rock shop; it asks the player to translate hexadecimal and octal values to decimal. In exchange for right answers, the player gets fuel (which is needed to advance) or money, depending on if they need fuel or not. Elfs are on all levels, they will ask a simple multiplication, addition, subtraction, or division question, and will reward the player with money, a bonus was added in version 2 for multiplication and division. Elfs with scrolls will also give the player a clue to help them find treasures. Stores are the same on all levels, players can but either nets or liters of fuel. Looking for treasures is the same on all levels. The player will be asked to stack a number of snowballs (spelled snowbols in the game) and then asked if they want to search with that stack number or revise the stack number. This is where the clue from the scroll elf helps, the player cannot find treasures unless they have the right stack number, and looking for treasures costs money, there are only 5 treasures to collect on each visit to a level. Finally, when the player has 4 liters of fuel, then they can ascend to the next floor, they must have 4 liters of fuel to do this, they are not required to collect treasures before ascending. If a player ascends on the third level, they will deposit the treasures they have collected and return to the first level to repeat the process. The goal is to collect 200 treasures which the player can now do at their lisure since they cannot exit the game without saving their progress, and there is a method in place to load the player’s data into the next running of the program so that they may pick up where they left off.

Summary

Project size: 600+ lines

Number of variables: 35

Number of arrays: 1 (size of 6)

Number of functions: 8

Number of io files: 2

Each activity has a function, except for ascension; many of them reference variables in the main function. Mostly random functions and tests to see if the player has enough, money, fuel, treasures ect. Uses the mathematics operators extensively, as questions are generated randomly and their answers are calculated by the computer. Do while loops are employed to repeat processes until a certain action is taken. While I was building the game individually the same variables would appear each time, but I used the level counter to change it, version 2 employed ctime which helped randomize the questions. Switch statements are used, to read information for the right level, and for the actions a player can take on each level. Moving the player’s stats into an array made them easier to pass into a function. I also tried to make the save and load routines into functions but could not figure out how to pass in the ifstreams and ofstreams in as objects. I also realized that each time a file was opened, it destroyed the data that was in it that was the reason for it’s opening. This is what caused the crude system of saving and loading the player’s data, since it does not take much to realize how one could cheat, and write in whatever numbers they want. But doing so would miss the point of increasing mathematic skill, as cheating wouldn’t help this otherwise boring and difficult math game.

Description

The point of this game is to make the player more adept at metal computations. I would call it an educational game, but it really just focuses on simple exercises rather than complex problem solving. The problems are intended to hard enough to require some thought (especially the Rockshop) but easy enough as to where the player would use scratch paper at first and eventually try to move away from it as their mind adjusted. This also accustoms the player to computations involved in translating number bases. The player is told that there are 200 treasures that they need to deposit, this was increased since players now have ht ability to save and load their game data, and can be done in as little as 14 ascents to level 3. The ending explains that the treasures are really a metaphor for mathematical skill implying that mathematical skill itself is a treasure to be labored for.

Flowchart

Too large to fit in one 8.5X11in page, see folder contents

Pseudo Code

Libraries

Function prototypes

Declare variables, initialize some of them

Output title story synopsis, ask player if they want the instructions

Input answer

If yes, output instructions,

If no, output “off we go then”

Do while loop for entire game, as long as treasures deposited is less than 150

Calculate the level the player is on, (returns a 1, 2, or 3)

Switch based on floor calculation

if case 1

randomly generate number of snowballs needed for this visit to find treasures

set number of treasures available equal to 5

do while loop for level 1

output menu options for the player

ask player if they want to load game or start new

input response,

if statement based on response,

if player loads then the array for their stats is read from a file

if player restarts, then array is filled with predetermined beginning values

level counter initializes to 3

treasures deposited initializes at 0

start do while loop for entire game

calculate what floor the player Is on by moding the level counter by three and adding one

switch based on floor number (either 1 2 or 3)

case1 is the code for level one

set number of available treasures to 5

randomly generate a number for the stack of snowballs

start a do while loop until player increases the value of the level counter

output the player’s options of where to go

input response

another switch statement based on the response

case 1 calls elf w/ scroll function. References stats array, and the stack number

output money, nets, fuel, and treasures found when function finishes

case2 calls elf w/out scroll function, references stats array only

output money, nets, fuel, and treasures when function finishes

case3 calls clockshop function, references fuel and money

output money, nets, fuel, and treasures when function finishes

case4 calls store function, references money, nets and fuel

output money, nets, fuel, and treasures when function finishes

case5 calls treasure search function, references money, number of snowballs needed

output money, nets, fuel, and treasures when function finishes

case 6 test if play has 4 liters of fuel

if true add one to level counter, subtract 4 from fuel counter

else tell player to get more fuel

output money, nets, fuel, and treasures

case 7 saves players stats in a file, asks player if they want to leave

if statement based on response,

if player wants to leave then program closes files and exits

else player is reminded to save before exiting

default outputs invalid option

close do while loop for level one test to see if level counter mod3 plus one is equal to one

break case1 for floor switch

case 2 is the code for level two

set number of available treasures to 5

randomly generate a number for the stack of snowballs

start a do while loop until player increases the value of the level counter

output the player’s options of where to go

input response

another switch statement based on the response

case 1 calls elf w/ scroll function. References stats array, and the stack number

output money, nets, fuel, and treasures found when function finishes

case2 calls elf w/out scroll function, references stats array only

output money, nets, fuel, and treasures when function finishes

case3 calls scaleshop function, references fuel and money

output money, nets, fuel, and treasures when function finishes

case4 calls store function, references money, nets and fuel

output money, nets, fuel, and treasures when function finishes

case5 calls treasure search function, references money, number of snowballs needed

output money, nets, fuel, and treasures when function finishes

case 6 test if play has 4 liters of fuel

if true add one to level counter, subtract 4 from fuel counter

else tell player to get more fuel

output money, nets, fuel, and treasures

case 7 saves players stats in a file, asks player if they want to leave

if statement based on response,

if player wants to leave then program closes files and exits

else player is reminded to save before exiting

default outputs invalid option

close do while loop for level two test if level counter mod3 plus one is equal to two

break case2 for floor switch

case 3 is the code for level three

set number of available treasures to 5

randomly generate a number for the stack of snowballs

start a do while loop until player increases the value of the level counter

output the player’s options of where to go

input response

another switch statement based on the response

case 1 calls elf w/ scroll function. References stats array, and the stack number

output money, nets, fuel, and treasures found when function finishes

case2 calls elf w/out scroll function, references stats array only

output money, nets, fuel, and treasures when function finishes

case3 calls scaleshop function, references fuel and money

output money, nets, fuel, and treasures when function finishes

case4 calls store function, references money, nets and fuel

output money, nets, fuel, and treasures when function finishes

case5 calls treasure search function, references money, number of snowballs needed

output money, nets, fuel, and treasures when function finishes

case 6 test if play has 4 liters of fuel

if true add one to level counter, subtract 4 from fuel counter

run treasure deposit function calls stats array;

else tell player to get more fuel

output money, nets, fuel, and treasures

case 7 saves players stats in a file, asks player if they want to leave

if statement based on response,

if player wants to leave then program closes files and exits

else player is reminded to save before exiting

default outputs invalid option

close do while loop for level three test if level counter mod3 plus one is equal to three

break case3 for floor switch

end do while loop for entire game, test for number of treasures that were deposited

Close bracket for do while loop for the game test to see if treasures is less than 200

if true then execute the loop again

else Output end sequence

End main

Major variables

|  |  |  |  |
| --- | --- | --- | --- |
| type | name | description | Location |
| char | instruct | Where player inputs answer when asked if he or she wants to see instructions | Main |
| char | decide | used to hold player’s response after they are asked if they want to load a game or start a new one | Main |
| int | gohtoo | Where player inputs choice of what action he or she want to take on a level | Main |
| int | nnetz | Holds number of net items that the player has, player start with 3 nets | Main |
| int | floor | Switch statement to determine what level the player is on is based on this variable, is result of computation done on the level counter | Main |
| int | money | Holds how much money the player has in cents, player starts off with 100 | Main |
| int | tresfnd | Holds how many treasures the player has found for each round, designed not to exceed 15 | Main |
| int | tresdep | Holds how many treasures the player deposited at the top, so that player can collect more in the next round, is tested to see if it is less than 150 to continue or end the game | Main |
| int | nsnblws | Stores randomly generated number for the number of snowballs the player needs to stack for that visit to that level, changes at the start of each level | Main |
| int | nliterz | Hold how many liters of fuel the player has, designed not to exceed 4 | Main |
| int | treserz | Set to 5 at the start of each level to restrict the number of treasures that can be found each visit to each floor |  |
| Unsigned int | levelct | Counter to calculate what level the player is on, the value keeps going up, but computation done on it only returns 1,2,3 also used to further randomize questions | Main |
| int | rspns | Use as input for player’s answer to an elf question | elfgets,  elfgetm |
| int | answer | Used to store answer to the elf question, tested against rspns to see if player is correct or not | elfgets,  elfgetm |
| int | operand | Used to determine what operation the question will be with %4 | elfgets,  elfgetm |
| int | numb1 | Random number between 0 and 333 to be used in elf question | elfgets,  elfgetm |
| int | numb2 | Random number between 0 and 333 to be used in elf question | elfgets,  elfgetm |
| int | hr1 | Used as the hour reading for first clock | clkshop |
| int | min1 | used as minute reading for first clock | clkshop |
| int | hr2 | Used as the hour reading for second clock | clkshop |
| int | min2 | used as minute reading for second clock | clkshop |
| int | mintime1 | used to put 1st clock time into a comparable value | clkshop |
| int | mintime2 | used to put 2nd clock time into a comparable value | clkshop |
| int | respmint | puts players response into comparable value | clkshop |
| int | ansmint | comparable value of the answer to clock question | clkshop |
| int | resphr | player inputs hour answer into this variable | clkshop |
| int | respmin | player inputs minute answer into this variable | clkshop |
| char | exit | used for player to exit the current shop | clkshop,  sclshop,  rockshop |
| int | rokmass | randomly generated number between 1 and 30 that player must guess and check for | sclshop |
| int | guess | determined how much a player wants to add to their previous guess, input a negative number to subtract | sclshop |
| int | gestotl | total of how much weight the player has added or subtracted | sclshop |
| int | hoctex | randomly generated number to be transcribed to octal or hex and then shown to player and compaired to palyer response | rokshop |
| int | decide | random variable that decides if hoctex will be shown as an octal or hex value to the player | rokshop |
| int | answer | where player inputs answer to be tested against the random translated number | rokshop |
| char | choice | used for player in a switch to determine what to buy at a store | storeget |
| char | done | where player inputs response to either exit or remain at a store | storeget |
| int | stack | number of snowballs that play inputs, must match nsnblws if player is to find treasures | tresrget |
| char | check | used in a switch, asks if player wants to revise their stack number, | tresrget |

Arrays:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| type | name | index | description | location |
| int | statz | 0 | holds the number of nets player has |  |
| int | statz | 1 | holds amount of money player has |  |
| int | statz | 2 | holds the number of treasures the player has found |  |
| int | statz | 3 | holds the number of treasures the player has deposited |  |
| int | statz | 4 | holds the number liters of jetpack fuel |  |
| int | statz | 5 |  |  |

Constructs used

|  |  |  |
| --- | --- | --- |
| Chapter | Construct | Location |
| 2 | int, char,unsigned int | iostream |
| 2 | cin,cout | iostream |
| 1 | +,-,/,\*,%,= | iostream |
| 1 | // | iostream |
| 2 | ==,>,<,>=,<=,!= | iostream |
| 3 | do{}while() | iostream |
| 2 | if,else | iostream |
| 3 | switch(), case:, break; | iostream |
| 3 | &&,||, | iostream |
| ?? | <<hex<<,<<oct<< | iostream |
| 3,4 | rand(),srand() | cstdlib |
| 4 | function prototypes, call, definitions, call by value | iostream |
| 6 | writing to a file | fstream |
| 7 | arrays | iostream |
| 4 | time as a random seed | ctime |

Reference

RCC CIS/CSC lab staff

<http://en.wikipedia.org/wiki/Treasure_MathStorm!> (note that my memory confirms this)

*Problem Solving with C++*8th edition, Walter Stavitch

Program

/\*

Name: Main.cpp

Copyright:

Author: Oliver D. Najera

Date: 18-11-2013 17:26

Description: version 2 of TreasureGlacier!

\*/

#include <cstdlib>

#include <iostream>

#include<fstream>

#include<ctime>

using namespace std;

//function prototypes

int elfgets(int[],int);

//int elfgets(int&,int&,int,int);//nets,money,levelcounter,nsnowballs

int elfgetm(int[]);

//int elfgetm(int&,int&,int,int);//nets,money,levelcounter,nsnowballs

int clkshop(int[]);

//int clkshop(int&,int&,int);//money,fuel,levelcounter

int sclshop(int[]);

//int sclshop(int&,int&,int);//money,fuel,levelcounter

int rokshop(int[]);

//int rokshop(int&,int&,int);//money,fuel,levelcounter

int storget(int[]);

//int storeget(int&,int&,int&);//money,nets,fuel

int tresget(int[],int,int&);

//int tresrget(int&,int&,int,int&);//money,treasures found,nsnowballs,treasures remaining

//void savgame(int[],int);

//void lodgame(int[],int);

void deptres(int[]);

//void castwal(int[]);

int main(int argc, char \*argv[]){//execution begins here

ifstream in;

ofstream out;

out.open("savefile.dat");

in.open("savefile");

//declare variables

char instruc,decide;

int statz[6];

srand(static\_cast<int>(time(0)));

//(statz[0])------------------nets

//(statz[1])-----------------money

//(statz[2])-------treasures found

//(statz[3])-treasures depositited

//(statz[4])--------liters of fuel

//(statz[5])---------level counter

int gohtoo,floor,nsnblws,treserz; //declare variables

statz[5]=3;

statz[3]=0;

cout<<"TREASURE GLACIER!\*//copyright 2013\n"

<<"\*a non-profit fan based sequel to Treasure Mathstorm!\n";

cout<<"Now entering Tresure Mountian\n"

<<"once upon a time there was a mountian called 'Treasure Mounitan' \n"

<<"It was named that because the treasures that were stored at the Mountiantop\n"

<<"Palace ensured that the surrounding countryside was prosperous, peaceful\n"

<<"and soverign. However there was this jerk, Morty Maxwell, who wanted\n"

<<"control of the contryside for himself. One day he came in to the palace and\n"

<<"stole all the treasures and scattered them all over Treasure Mountian\n"

<<"to make sure that the treasures were not returned to the Mountiantop Palace\n"

<<"he froze the mountian solid, and he put a spell on the inhabiting elves.\n"

<<"this spell gave them no desire to find or return the treasures, and it\n"

<<"made them speak in math problems so that they wouldn't help anyone\n"

<<"since the humans and wizards in the surrpunding contryside, were not good\n"

<<"at mountian climbing or math problems, it would take a hero from outside to\n"

<<"find the treasures, and bring them back to the Mountiantop Palace. That hero\n"

<<"will be you, the user, The fate of Treasure Mountian, Shady Glenn(the name\n"

<<"of the countryside), and all of the inhabitants rest in your hands. The\n"

<<"because the presence of a frozen mountian has introduced a perpetual winter\n"

<<"that only you have the skills to fix...\n"

<<"is this your first time playing? if so, then type a 'y' to show instructions\n"

<<"If you know how to play then type 'n' to begin\n";

cin>>instruc;

if (instruc=='y'){

cout<<"Alright, Well Trasure Glacier is separated into three levels\n"

<<"in each level there are five treasures to collect\n"

<<"to find the treasures you need to find an elf with a scroll\n"

<<"elfs will ask you a math question when you catch them\n"

<<"asnwer the question wrong and they will leave you nothing\n"

<<"get the question right and that elf will pay you in cash\n"

<<"elfs with the scroll will pay you and give the clue for that level\n"

<<"the clue is a number btween 0 and 9 refers to how many snowballs you\n"

<<"need to stack on the treasure terminal to find the treasures\n"

<<"However stacking the right number of snowballs is not enough\n"

<<"you must also throw down a coin, so treasures will cost money.\n"

<<"To catch elfs you need nets. Because elfs rip out of nets,\n"

<<"nets can only be used once. you can purcase more at a store\n"

<<"there is a store on each level; they sell nets and jetpack fuel\n"

<<"Jetpack fuel is how you advance to the next level\n"

<<"fuel sells in increments of 1 liter\n"

<<"you will need 4 liters of fuel to move up one level\n"

<<"however stores are not the only place to get fuel, nor are elves\n"

<<"the only way to get moeny.\n"

<<"shops will give you jetpack fuel for solving their problems\n"

<<"you get one liter for one right answer\n"

<<"if you asnswer more than 4 questions, the shops will give you money\n"

<<"don't confuse shops with stores\n";

system("PAUSE");

cout<<"there are three different shops, one for each level\n"

<<"the clockshop will ask you to sycronize clocks\n"

<<"the scaleshop will ask you to balance scales\n"

<<"the rockshop will ask you to match numbers in different bases\n"

<<"elves will ask mutliplication division\n"

<<"addition and subtraction\n"

<<"there are 200 treasures to collect, and you can only carry\n"

<<"so many, so you will have to make the acent many times,\n"

<<"acending to the next level on the third level will deposit the treasures\n"

<<"that you have collected in the caslte and put you back on the first level\n"

<<"it is best to exit once you have saved your progress\n";

}

if (instruc=='n'){

cout<<"off we go then!\n";}//if instruct==n, close

cout<<"do you want to continue where you left off, or start anew?\n"

<<"press 'n' for a new game, anything else to load, do not load if there is not\n"

<<"a save file\n";

cin>>instruc;

if(instruc=='n'){

statz[0]=3;

statz[1]=150;

statz[2]=0;

statz[3]=0;

statz[4]=2;

statz[5]=3;

}

else{

//lodgame(statz,in,6);

in>>statz[0]

>>statz[1]

>>statz[2]

>>statz[3]

>>statz[4]

>>statz[5];

cout<<"progress loaded\n";

}

do{//do while loop for the entire game, stops when player wins

floor=statz[5]%3+1;

//floor=levelct%3+1;

switch(floor){

case 1:{// for level one

nsnblws=(rand()%10+1);

treserz=5;

//savgame(statz,6);//autosave players data at the beggining of each floor

do{//start looping for level one

cout<<"you are on level 1, where do you wish to go?\n"

<<"type 1 to go catch an elf with a scroll\n"

<<"type 2 to go catch an elf\n"

<<"type 3 to go to the Clockshop\n"

<<"type 4 to go to the store\n"

<<"type 5 to look for treasures\n"

<<"type 6 to ascend to the next level\n"

<<"type 7 to save your progress\n";

cin>>gohtoo;

switch(gohtoo){

case 1:

elfgets(statz,nsnblws);

//elfgets(nnetz,money,levelct,nsnblws);//player wants to catch an elf with a scroll

cout<<"|money: "<<statz[1]<<"|nets: "<<statz[0]<<"|fuel: "<<statz[4]<<"|treasures: "<<statz[2]<<endl;//display player stats

//cout<<"|money: "<<money<<"|nets: "<<nnetz<<"|fuel: "<<nliterz<<"|treasures: "<<tresfnd<<endl;//display player stats

break;

case 2:{

elfgetm(statz);

//elfgetm(nnetz,money,levelct,nsnblws);

cout<<"|money: "<<statz[1]<<"|nets: "<<statz[0]<<"|fuel: "<<statz[4]<<"|treasures: "<<statz[2]<<endl;//display player stats

//cout<<"|money: "<<money<<"|nets: "<<nnetz<<"|fuel: "<<nliterz<<"|treasures: "<<tresfnd<<endl;//display player stats

}

break;

case 3:{

clkshop(statz);//player goes to clockshop

//clkshop(money,nliterz,levelct);//player goes to clockshop

cout<<"|money: "<<statz[1]<<"|nets: "<<statz[0]<<"|fuel: "<<statz[4]<<"|treasures: "<<statz[2]<<endl;//display player stats

//cout<<"|money: "<<money<<"|nets: "<<nnetz<<"|fuel: "<<nliterz<<"|treasures: "<<tresfnd<<endl;//display player stats

}

break;

case 4:{

storget(statz);//player goes to store

//storeget(money,nnetz,nliterz);//player goes to store

cout<<"|money: "<<statz[1]<<"|nets: "<<statz[0]<<"|fuel: "<<statz[4]<<"|treasures: "<<statz[2]<<endl;//display player stats

//cout<<"|money: "<<money<<"|nets: "<<nnetz<<"|fuel: "<<nliterz<<"|treasures: "<<tresfnd<<endl;//display player stats

}

break;

case 5:{

tresget(statz,nsnblws,treserz);//player goes to find treasures

//tresrget(money,tresfnd,nsnblws,treserz);//player goes to find treasures

cout<<"|money: "<<statz[1]<<"|nets: "<<statz[0]<<"|fuel: "<<statz[4]<<"|treasures: "<<statz[2]<<endl;//display player stats

//cout<<"|money: "<<money<<"|nets: "<<nnetz<<"|fuel: "<<nliterz<<"|treasures: "<<tresfnd<<endl;//display player stats

}

break;

case 6:{//player moves to next level

if(statz[4]>=4){

//if(nliterz>=4){

cout<<"at last, the 20kg on your back finds its niche, as you quickly make your acent\n"

<<"to the next level\n";

statz[4]-=4;

//nliterz-=4;

statz[5]+=1;

//levelct+=1;

//savgame(statz,6);//autosave players data at the exit of each floor

}

else{

cout<<"you do not have enough fuel, go to a shop or buy some at the store\n";

}

cout<<"|money: "<<statz[1]<<"|nets: "<<statz[0]<<"|fuel: "<<statz[4]<<"|treasures: "<<statz[2]<<endl;//display player stats

//cout<<"|money: "<<money<<"|nets: "<<nnetz<<"|fuel: "<<nliterz<<"|treasures: "<<tresfnd<<endl;//display player stats

}

break;

case 7:{

//savgame(statz,out,in,6);

out<<statz[0]<<endl

<<statz[1]<<endl

<<statz[2]<<endl

<<statz[3]<<endl

<<statz[4]<<endl

<<statz[5]<<endl;

cout<<"progress saved\n";

cout<<"would you like to exit the application?\n"

<<"input 'y' to leave, anyhting else to keep playing\n"

<<"when you want to load your information, copy the data from savefile.dat\n"

<<"and paste it into savefile (of type file), then it will load correctly\n";

cin>>decide;

if((decide=='y')||(decide=='Y')){

cout<<"goodbye\n";

out.close();

in.close();

system("PAUSE");

exit(0);

}

else

cout<<"okay then, just make sure to save when you do want to exit\n";

}

break;

default:{//player cannot follow intructions

cout<<"that is not an option\n";

cout<<"|money: "<<statz[1]<<"|nets: "<<statz[0]<<"|fuel: "<<statz[4]<<"|treasures: "<<statz[2]<<endl;//display player stats

//cout<<"|money: "<<money<<"|nets: "<<nnetz<<"|fuel: "<<nliterz<<"|treasures: "<<tresfnd<<endl;//display player stats

}

break;

}//switch for level 1 options

}while((statz[5]%3+1)==1);

//}while((levelct%3+1)==1);

break;

}//end of level one case information

case 2:{//for level two,

nsnblws=(rand()%10+1);

treserz=5;

//savgame(statz,6);//autosave players data at the beggining of each floor

do{

cout<<"you are on level 2, where do you wish to go?\n"

<<"type 1 to go catch an elf with a scroll\n"

<<"type 2 to go catch an elf\n"

<<"type 3 to go to the Scaleshop\n"

<<"type 4 to go to the store\n"

<<"type 5 to look for treasures\n"

<<"type 6 to ascend to the next level\n"

<<"type 7 to save your progress\n";

cin>>gohtoo;

switch(gohtoo){

case 1:

elfgets(statz,nsnblws);

//elfgets(nnetz,money,levelct,nsnblws);//player wants to catch an elf with a scroll

cout<<"|money: "<<statz[1]<<"|nets: "<<statz[0]<<"|fuel: "<<statz[4]<<"|treasures: "<<statz[2]<<endl;//display player stats

//cout<<"|money: "<<money<<"|nets: "<<nnetz<<"|fuel: "<<nliterz<<"|treasures: "<<tresfnd<<endl;//display player stats

break;

case 2:{

elfgetm(statz);

//elfgetm(nnetz,money,levelct,nsnblws);

cout<<"|money: "<<statz[1]<<"|nets: "<<statz[0]<<"|fuel: "<<statz[4]<<"|treasures: "<<statz[2]<<endl;//display player stats

//cout<<"|money: "<<money<<"|nets: "<<nnetz<<"|fuel: "<<nliterz<<"|treasures: "<<tresfnd<<endl;//display player stats

}

break;

case 3:{

sclshop(statz);//player goes to scaleshop

//sclshop(nliterz,money,levelct);//player goes to scaleshop

cout<<"|money: "<<statz[1]<<"|nets: "<<statz[0]<<"|fuel: "<<statz[4]<<"|treasures: "<<statz[2]<<endl;//display player stats

//cout<<"|money: "<<money<<"|nets: "<<nnetz<<"|fuel: "<<nliterz<<"|treasures: "<<tresfnd<<endl;//display player stats

}

break;

case 4:{

storget(statz);//player goes to store

//storeget(money,nnetz,nliterz);//player goes to store

cout<<"|money: "<<statz[1]<<"|nets: "<<statz[0]<<"|fuel: "<<statz[4]<<"|treasures: "<<statz[2]<<endl;//display player stats

//cout<<"|money: "<<money<<"|nets: "<<nnetz<<"|fuel: "<<nliterz<<"|treasures: "<<tresfnd<<endl;//display player stats

}

break;

case 5:{

tresget(statz,nsnblws,treserz);//player goes to find treasures

//tresrget(money,tresfnd,nsnblws,treserz);//player goes to find treasures

cout<<"|money: "<<statz[1]<<"|nets: "<<statz[0]<<"|fuel: "<<statz[4]<<"|treasures: "<<statz[2]<<endl;//display player stats

//cout<<"|money: "<<money<<"|nets: "<<nnetz<<"|fuel: "<<nliterz<<"|treasures: "<<tresfnd<<endl;//display player stats

}

break;

case 6:{//player moves to next level

if(statz[4]>=4){

//if(nliterz>=4){

cout<<"at last, the 20kg on your back finds its niche, as you quickly make your acent\n"

<<"to the next level\n";

statz[4]-=4;

//nliterz-=4;

statz[5]+=1;

//levelct+=1;

//savgame(statz,6);//autosave players data at the exit of each floor

}

else{

cout<<"you do not have enough fuel, go to a shop or buy some at the store\n";

}

cout<<"|money: "<<statz[1]<<"|nets: "<<statz[0]<<"|fuel: "<<statz[4]<<"|treasures: "<<statz[2]<<endl;//display player stats

//cout<<"|money: "<<money<<"|nets: "<<nnetz<<"|fuel: "<<nliterz<<"|treasures: "<<tresfnd<<endl;//display player stats

}

break;

case 7:{

//savgame(statz,out,in,6);

out<<statz[0]<<endl

<<statz[1]<<endl

<<statz[2]<<endl

<<statz[3]<<endl

<<statz[4]<<endl

<<statz[5]<<endl;

cout<<"progress saved\n";

cout<<"would you like to exit the application?\n"

<<"input 'y' to leave, anyhting else to keep playing\n"

<<"when you want to load your information, copy the data from savefile.dat\n"

<<"and paste it into savefile (of type file), then it will load correctly\n";

cin>>decide;

if((decide=='y')||(decide=='Y')){

cout<<"goodbye\n";

out.close();

in.close();

system("PAUSE");

exit(0);

}

else

cout<<"okay then, just make sure to save when you do want to exit\n";

}

break;

default:{//player cannot follow intructions

cout<<"that is not an option\n";

cout<<"|money: "<<statz[1]<<"|nets: "<<statz[0]<<"|fuel: "<<statz[4]<<"|treasures: "<<statz[2]<<endl;//display player stats

//cout<<"|money: "<<money<<"|nets: "<<nnetz<<"|fuel: "<<nliterz<<"|treasures: "<<tresfnd<<endl;//display player stats

}

break;

}//switch for level 2 options

}while((statz[5]%3+1)==2);

//}while((levelct%3+1)==2);

break;

}//case2

break;

case 3:{

nsnblws=(rand()%10+1);

treserz=5;

//savgame(statz,6);//autosave players data at the beggining of each floor

do{

cout<<"you are on level 3, where do you wish to go?\n"

<<"type 1 to go catch an elf with a scroll\n"

<<"type 2 to go catch an elf\n"

<<"type 3 to go to the Rockshop\n"

<<"type 4 to go to the store\n"

<<"type 5 to look for treasures\n"

<<"type 6 to ascend to the next level\n"

<<"type 7 to save your progress\n";

cin>>gohtoo;

switch(gohtoo){

case 1:

elfgets(statz,nsnblws);

//elfgets(nnetz,money,levelct,nsnblws);//player wants to catch an elf with a scroll

cout<<"|money: "<<statz[1]<<"|nets: "<<statz[0]<<"|fuel: "<<statz[4]<<"|treasures: "<<statz[2]<<endl;//display player stats

//cout<<"|money: "<<money<<"|nets: "<<nnetz<<"|fuel: "<<nliterz<<"|treasures: "<<tresfnd<<endl;//display player stats

break;

case 2:{

elfgetm(statz);

//elfgetm(nnetz,money,levelct,nsnblws);

cout<<"|money: "<<statz[1]<<"|nets: "<<statz[0]<<"|fuel: "<<statz[4]<<"|treasures: "<<statz[2]<<endl;//display player stats

//cout<<"|money: "<<money<<"|nets: "<<nnetz<<"|fuel: "<<nliterz<<"|treasures: "<<tresfnd<<endl;//display player stats

}

break;

case 3:{

rokshop(statz);//player goes to rockshop

//rokshop(nliterz,money,levelct);//player goes to scaleshop

cout<<"|money: "<<statz[1]<<"|nets: "<<statz[0]<<"|fuel: "<<statz[4]<<"|treasures: "<<statz[2]<<endl;//display player stats

//cout<<"|money: "<<money<<"|nets: "<<nnetz<<"|fuel: "<<nliterz<<"|treasures: "<<tresfnd<<endl;//display player stats

}

break;

case 4:{

storget(statz);//player goes to store

//storeget(money,nnetz,nliterz);//player goes to store

cout<<"|money: "<<statz[1]<<"|nets: "<<statz[0]<<"|fuel: "<<statz[4]<<"|treasures: "<<statz[2]<<endl;//display player stats

//cout<<"|money: "<<money<<"|nets: "<<nnetz<<"|fuel: "<<nliterz<<"|treasures: "<<tresfnd<<endl;//display player stats

}

break;

case 5:{

tresget(statz,nsnblws,treserz);//player goes to find treasures

//tresrget(money,tresfnd,nsnblws,treserz);//player goes to find treasures

cout<<"|money: "<<statz[1]<<"|nets: "<<statz[0]<<"|fuel: "<<statz[4]<<"|treasures: "<<statz[2]<<endl;//display player stats

//cout<<"|money: "<<money<<"|nets: "<<nnetz<<"|fuel: "<<nliterz<<"|treasures: "<<tresfnd<<endl;//display player stats

}

break;

case 6:{//player moves to next level

if(statz[4]>=4){

//if(nliterz>=4){

cout<<"at last, the 20kg on your back finds its niche, as you quickly make your acent\n"

<<"to the next level\n";

statz[4]-=4;

//nliterz-=4;

statz[5]+=1;

//levelct+=1;

deptres(statz);

//savgame(statz,6);//autosave players data at the exit of each floor

}

else{

cout<<"you do not have enough fuel, go to a shop or buy some at the store\n";

}

cout<<"|money: "<<statz[1]<<"|nets: "<<statz[0]<<"|fuel: "<<statz[4]<<"|treasures: "<<statz[2]<<endl;//display player stats

//cout<<"|money: "<<money<<"|nets: "<<nnetz<<"|fuel: "<<nliterz<<"|treasures: "<<tresfnd<<endl;//display player stats

}

break;

case 7:{

//savgame(statz,out,in,6);

out<<statz[0]<<endl

<<statz[1]<<endl

<<statz[2]<<endl

<<statz[3]<<endl

<<statz[4]<<endl

<<statz[5]<<endl;

cout<<"progress saved\n";

cout<<"would you like to exit the application?\n"

<<"input 'y' to leave, anyhting else to keep playing\n"

<<"when you want to load your information, copy the data from savefile.dat\n"

<<"and paste it into savefile (of type file), then it will load correctly\n";

cin>>decide;

if((decide=='y')||(decide=='Y')){

cout<<"goodbye\n";

out.close();

in.close();

system("PAUSE");

exit(0);

}

else

cout<<"okay then, just make sure to save when you do want to exit\n";

}

break;

default:{//player cannot follow intructions

cout<<"that is not an option\n";

cout<<"|money: "<<statz[1]<<"|nets: "<<statz[0]<<"|fuel: "<<statz[4]<<"|treasures: "<<statz[2]<<endl;//display player stats

//cout<<"|money: "<<money<<"|nets: "<<nnetz<<"|fuel: "<<nliterz<<"|treasures: "<<tresfnd<<endl;//display player stats

}

break;

}//switch for level 3 options

//}while((levelct%3+1)==3);

}while((statz[5]%3+1)==3);//case 3

break;

}//level3floor switch

//}while(tresdep<=200);//end of the do while loop for the entire game

}//switch for the levels based on the level counter

}while(statz[3]<=200);//end of the do while loop for the entire game

cout<<"congradulations, you reached the end of a game which is really all about\n"

<<"getting as many treasures as possible before you got bored and left!\n"

<<"Watch out for TREASURE GLACIER0.1! for updates and paches\n";

cout<<"you, are truely patient or have an excellent feel for mathematics\n"

<<"if you have not realized yet, the 'treasure' was increased skill in mathematics\n"

<<"'And because in all the galaxy they had found nothing more precious than the mind,\n"

<<"they encouraged its dawning elsewhere' -(prologue; 3001 The Final Odyssey)\n"

<<"now go out and use your mathematic skill to solve some real problems\n";

system("PAUSE");

return EXIT\_SUCCESS;

}//end main

int elfgets(int stz[],int nsnblws){

//int elfgets(int &nnetz,int &money,int levelct,int nsnblws){//when an elf with scroll is caught//nnetz,money,levelct,nsnblws

cout<<"you caught an elf\n";

srand(static\_cast<int>(time(0)));

int rspns,answer,operand,numb3;

int numb1=((rand()\*stz[5])%333);

//int numb1=((rand()\*levelct)%333);

int numb2=rand()%333;

operand=(rand()+stz[5])%4+1;

//operand=(rand()+levelct)%4+1;

if(stz[0]>0){//test to see if the player has nets

//if (nnetz>0){//see if player has nets

stz[0]-=1;

//nnetz-=1;

switch(operand){

case 1:

answer=numb1+numb2;

cout<<"what is "<<numb1<<"+"<<numb2<<" ?\n";

cin>>rspns;

break;

case 2:

answer=numb1-numb2;

cout<<"what is "<<numb1<<"-"<<numb2<<" ?\n";

cin>>rspns;

break;

case 3:

answer=numb1\*numb2;

cout<<"what is "<<numb1<<"\*"<<numb2<<" ?\n";

cin>>rspns;

break;

case 4:

numb3=numb1\*numb2;

//answer=numb3/numb2;

answer=numb1;

cout<<"what is "<<numb3<<"/"<<numb2<<"\n";

cin>>rspns;

break;

}//operand switch

if (answer==rspns){//see if player was right

cout<<"great you got it, here's thirty cents,\n"

<<"the number of snowbols you need to stack is "<<nsnblws<<" snowbols\n";

stz[1]+=30;

//money+=30;

if ((operand==4)||(operand==3)){//bonus for multiplication problems

cout<<"bonus for multiplication and division\n";

stz[1]+=20;

//money+=20;

}

}

else//player was wrong

cout<<"that's not the answer stuipid. sorry I can't help yah\n";

}//senario for when player has more than one net

else{//if player does not have nets

stz[0]-=0;

//nnetz-=0;

cout<<"you'll need more nets than that to catch me!\n";

}

//cout<<"|money: "<<stz[1]<<"|nets: "<<stz[0]<<endl;

//cout<<"|money: "<<money<<"|nets: "<<nnetz<<endl;

}//end elfgets

int elfgetm(int stz[]){

//int elfgetm(int &nnetz,int &money,int levelct,int nsnblws){// for catching al elf without the scroll

int rspns,answer,numb3;

srand(static\_cast<int>(time(0)));

int numb1=(rand()\*stz[5])%333;

//int numb1=(rand()\*levelct)%333;

int numb2=rand()%333;

int operand=(stz[5]\*rand())%4+1;

if (stz[0]>0){

//if (nnetz>0){

stz[0]-=1;

//nnetz-=1;

switch(operand){

case 1:

answer=numb1+numb2;

cout<<"what is "<<numb1<<"+"<<numb2<<" ?\n";

cin>>rspns;

break;

case 2:

answer=numb1-numb2;

cout<<"what is "<<numb1<<"-"<<numb2<<" ?\n";

cin>>rspns;

break;

case 3:

answer=numb1\*numb2;

cout<<"what is "<<numb1<<"\*"<<numb2<<" ?\n";

cin>>rspns;

break;

case 4:

numb3=numb1\*numb2;

//answer=numb3/numb2;

answer=numb1;

cout<<"what is "<<numb3<<"/"<<numb2<<"\n";

cin>>rspns;

break;

}

if (answer==rspns){

cout<<"great you got it, here's thirty cents\n";

stz[1]+=30;

//money+=30;

if ((operand==4)||(operand==3)){//bonus for multiplication problems

cout<<"bonus for multiplication and division\n";

stz[1]+=20;

//money+=20;

}

}

else

cout<<"that's not the answer stuipid. sorry I can't help yah\n";

}

else

cout<<"you'll need more nets than that to catch me!\n";

cout<<" you now have "<<stz[1]<<" cents and "<<stz[0]<<" nets\n";

//cout<<" you now have "<<money<<" cents and "<<nnetz<<" nets\n";

}

int clkshop(int stz[]){//clockshop function

//int clkshop(int &money,int &nliterz,int levelct){//clockshop function

int hr1,min1,m1ntime,hr2,min2,m2ntime,rspmint,ansmint,resphr,respmin;

char exit;

do{

if(stz[5]<=6){//to avoid redundancy with game instructions

//if(levelct<=6){//to avoid redundancy with game instructions

cout<<"welcome to the clockshop, due to the freezing of treasure mountian\n";

}

cout<<"my clocks have fallen out of time I need you to reset them\n";

if(stz[5]<=6){

//if(levelct<=6){

cout<<"be aware that if the second time is less than the first, then your answer\n"

<<"will be negative, and you can input more than 60 minutes if you mess up the\n"

<<"hour response, \n";

}

hr1=rand()%24;

hr2=rand()%24;

min1=rand()%60;

min2=rand()%60;

m1ntime=hr1\*60+min1;

m2ntime=hr2\*60+min2;

if(min1<10)

cout<<"1st clock says... "<<hr1<<":0"<<min1<<endl;

else

cout<<"1st clock says... "<<hr1<<":"<<min1<<endl;

if(min2<10)

cout<<"2nd clock says... "<<hr2<<":0"<<min2<<endl;

else

cout<<"2nd clock says... "<<hr2<<":"<<min2<<endl;

ansmint=m2ntime-m1ntime;

cout<<"what's the difference in...\n";

cout<<"hours?---> ";cin>>resphr;

cout<<"minutes?-> ";cin>>respmin;

rspmint=resphr\*60+respmin;

if(ansmint==rspmint){

cout<<"thanks, I think you got it, here's something for yah\n";

if (stz[4]<4)

//if (nliterz<4)

stz[4]+=1;

//nliterz+=1;

else

stz[1]+=25;

//money+=25;

}

else

cout<<"nope, don't think so\n";

cout<<"money: "<<stz[1]<<",fuel: "<<stz[4]<<endl;

//cout<<"money: "<<money<<",fuel: "<<nliterz<<endl;

cout<<"you want to reset some more? type 'y' to stay, anything else to leave\n";

cin>>exit;

}while (exit=='y');

}

int sclshop(int stz[]){

//int sclshop(int &nliterz,int &money,int levelct){//scaleshop function

int rokmass,guess,gestotl=0;

char exit;

if (stz[5]<=6){

//if(levelct<=6){//to avoid redundancy with game instructions

cout<<"welcome to the Scaleshop!\n"

<<"since Treasure Mountian has become frozen\n"

<<"all my scales have become unblanced\n"

<<"since I balance random rocks I find with known masses\n"

<<"I cannot tell you what the mass of the object is\n"

<<"the scale will tell you greater than or less than\n"

<<"but don't worry all of the objects will be between 1kg and 30kg\n"

<<"to add wights just type the number of kgs you want to add\n"

<<"if you go over, then type a negative number to subtract masses\n"

<<"if your under, well then just type a normal number without a negative\n"

<<"yeah it's a guessing game, but you need money and I need balance, now go,\n";

}

do{//start loop so player is asked if they want saty or leave

rokmass=rand()%30;

do{//start loop for guessing the random number

cout<<"so what is your guess?\n";

cin>>guess;

//exit=guess;

gestotl+=guess;

if(gestotl==rokmass){

cout<<"balance\n"

<<"the mass of rock="<<gestotl<<"kg\n";

if(stz[4]<4){

//if(nliterz<4){

stz[4]+=1;

//nliterz+=1;

}

else

stz[1]+=25;

//money+=25;

}

if(gestotl>rokmass){

cout<<"mass of rock<"<<gestotl<<"kg\n";

}

if(gestotl<rokmass){

cout<<"mass of rock>"<<gestotl<<"kg\n";

}

}while(gestotl!=rokmass);

cout<<"|money: "<<stz[1]<<"|fuel: "<<stz[4]<<endl;//show plyer their related statz

//cout<<"money: "<<money<<"fuel: "<<nliterz<<endl;//show plyer their related statz

cout<<"do you want to balance another scale?\n"//ask player if they want to leave or exit

<<"press 'y' to balance, anything else to leave\n";

cin>>exit;

}while(exit=='y');//answer determines if the loop will happen again or not

}

int rokshop(int stz[]){

//int rokshop(int &nliterz,int &money,int levelct){

//declare varables

int hoctex,decide,answer;

char exit;

if(stz[5]<=6){//to avoid redundancy with game instructions

//if(levelct<=6){//to avoid redundancy with game instructions

cout<<"welcome to the Rockshop, this is the logic cave\n"

<<"this is where the numbers for different bases are stored for reference\n"

<<"unfortunately the freezing loosened the hex and octal reference\n"

<<"so what I need to do is imput the decimal form of the hex or octal number\n"

<<"that appears, these values will generally range from 1 to 1000 in decimal\n";

}

do{//start loop until player wants to leave

hoctex=rand()%1000;

decide=rand()%2;

if(decide==1){

cout<<"what is this hex number?\n"

<<hex<<hoctex<<endl;

}

if(decide==0){

cout<<"what is this octal number?\n"

<<oct<<hoctex<<endl;

}

do{//loop until player has the right answer

cin>>answer;

if(answer==hoctex){

cout<<"yeah that's it\n";

if(stz[4]<4)

//if(nliterz<4)

stz[4]+=1;

//nliterz+=1;

else

stz[1]+=25;

//money+=25;

}

else{

cout<<"I don't think that is right, try again\n";

}

}while(answer!=hoctex);//end loop when player has right answer

cout<<"|money: "<<dec<<stz[1]<<"|fuel: "<<dec<<stz[4]<<endl;

//cout<<"|money: "<<dec<<money<<"|fuel: "<<dec<<nliterz<<endl;//show player their stats

cout<<"you want to do another? type 'y' to go again, anthing else to leave\n";

cin>>exit;

}while(exit=='y');

}

int storget(int stz[]){//store function

//int storeget(int &money,int &nnetz,int &nliterz){//store function

//declare variables

char choice,done;

cout<<"jetpack fuel,clif bars,nets; you want it we got it\n"

<<"as long as you have enough money\n";

do{//start loop until player wants to leave

//display options, promt player for their choice

cout<<"what would you like to buy?\n"

<<"press 1 for liter of rocket fuel: 200 cents\n"

<<"press 2 for 1 elf sized net : 5 cents\n"

<<"press 3 for 5 elf sized nets : 25 cents\n";

cin>>choice;

switch(choice){

case '1':

if(stz[1]>=200){

//if(money>=200){

if(stz[4]<4){

//if(nliterz<4){

cout<<"purchased 1 liter of rocket fuel\n";

stz[1]-=200;

//money-=200;

stz[4]+=1;

//nliterz+=1;

cout<<"|money: "<<stz[1]<<"|nets: "<<stz[0]<<"|fuel: "<<stz[4]<<endl;

//cout<<"|money: "<<money<<"|nets: "<<nnetz<<"|fuel: "<<nliterz<<endl;

}

else{

cout<<"your fuel tank is full\n";

cout<<"|money: "<<stz[0]<<"|nets: "<<stz[1]<<"|fuel: "<<stz[4]<<endl;

//cout<<"|money: "<<money<<"|nets: "<<nnetz<<"|fuel: "<<nliterz<<endl;

}

}

else{

cout<<"nice try, but you do not have enough money\n";}

break;

case '2':

if(stz[1]>=5){

//if(money>=5){

cout<<"purchased 1 elf sized net\n";

stz[1]-=5;

//money-=5;

stz[0]+=1;

//nnetz+=1;

cout<<"|money: "<<stz[1]<<"|nets: "<<stz[0]<<"|fuel: "<<stz[4]<<endl;

//cout<<"|money: "<<money<<"|nets: "<<nnetz<<"|fuel: "<<nliterz<<endl;

}

else{

cout<<"nice try, but you do not have enough money\n";}

break;

case '3':

if(stz[1]>=25){

//if(money>=25){

cout<<"purchased 5 elf sized nets\n";

stz[1]-=25;

//money-=25;

stz[0]+=5;

//nnetz+=5;

cout<<"|money: "<<stz[1]<<"|nets: "<<stz[0]<<"|fuel: "<<stz[4]<<endl;

//cout<<"|money: "<<money<<"|nets: "<<nnetz<<"|fuel: "<<nliterz<<endl;

}

else{

cout<<"nice try, but you do not have enough money\n";}

break;

default:

cout<<"sorry we are out of those\n";

break;

}

//ask player if they want to stay or leave, prompt for response

cout<<"is that all? type 'y' or any key to leave, and 'n' to make another purchase\n";

cin>>done;

}while((done=='N')||(done=='n'));//loop for another perchase

cout<<"thanks for shopping at T.M. Sporting Goods come back if you need anything!\n";

}

int tresget(int stz[],int nsnblws,int &treserz){

//int tresrget(int &money,int &tresfnd,int nsnblws,int &treserz){

int stack;

char check;

cout<<"you are at the treasure terminal, there is a small stone grotto in \n"

<<"the rockface of this side of Treasure Mountian. It is half buried in snow\n"

<<" and there is eveidence of a spring nearby which has now frozen.\n";

cout<<"how many snowbols do you wish to stack?\n";

cin>>stack;

do{// start loop until player wants to leave

cout<<"search for a treasure? Type 'y' to search, 's' to change stack number\n"

<<"and anything else to exit\n";

cin>>check;

switch(check){

case 'Y':

case 'y'://player wants to check for a treasure

if(treserz>0){// test to see if all treasures have been collected,

if(stack==nsnblws){//test to see if the player stacked the right number of snowbols

if(stz[1]>10){//check to see if player can spare a dime

//if(money>10){//check to see if player can spare a dime

cout<<"you find a treasure\n";

stz[2]+=1;

//tresfnd+=1;

treserz-=1;

stz[1]-=10;

//money-=10;

cout<<"|money = "<<stz[1]<<"|treasures = "<<stz[2]<<endl;

//cout<<"|money = "<<money<<"|treasures = "<<tresfnd<<endl;

}

else{//player cannot spare a dime

cout<<"you need more money\n";

cout<<"|money = "<<stz[1]<<"|treasures = "<<stz[2]<<endl;

//cout<<"|money = "<<money<<"|treasures = "<<tresfnd<<endl;

}

}

else{//player did not stack the right number of snowbols

cout<<"you did not stack the correct number of snowbols\n";

stz[1]-=10;

//money-=10;

}

}

else//player has extracted the maximun of treasures for this round

cout<<"you have found all the currently avalible treasures\n";

break;

case 'S':

case 's'://player wants to revise the stack

cout<<"how many do you wish to stack?\n";

cin>>stack;

break;

default://player wants to exit,

cout<<"you leave the grotto, still confused as to why you need to throw down a coin.\n";

break;

}

}while(((check=='S')||(check=='s'))||((check=='Y')||(check=='y')));//repeat until player askes to exit

cout<<"you are now back on the trail\n";

}

void deptres(int stz[]){

cout<<"your treasures are put back into the chest they once belonged to\n"

<<"this also frees up more space in your sachel\n";

stz[3]+=stz[2];

//cout<<"this also frees up more space in your sachel\n";

stz[2]=0;

cout<<"|treasures depositied: "<<stz[3]<<"|treasures found: "<<stz[2]<<endl;

}

/\*thingsto do

//fix clockshop,

//make playerstats a saveable file,

//remove redundant instructins,

//make player stats an array,

//increase reward for multiplication or devision,

add boss?

add theifs?

\*/

//just to even it out to a nice 925