1. Implement a "guess the number" game using a while loop. The program should randomly select a number between 1 and 100, and then it prompts the user to guess the number. The loop continues until the user guesses the number correctly, providing hints ("too high" or "too low") after each guess.

**CODE:-**

#include <iostream>

#include <cstdlib>

#include <ctime>

using namespace std;

int main()

{

int num, guess, tries = 0;

srand(time(0));

num = rand() % 100 + 1;

cout << "Guess My Number Game\n\n";

do

{

cout << "Enter a guess between 1 and 100 : ";

cin >> guess;

tries++;

if (guess > num)

cout << "Too high!\n\n";

else if (guess < num)

cout << "Too low!\n\n";

else

cout << "\nCorrect! You got it in " << tries << " guesses!\n";

} while (guess != num);

return 0;

}

**Output:-**

Guess My Number Game

Enter a guess between 1 and 100 : 34

Too low!

Enter a guess between 1 and 100 : 55

Too low!

Enter a guess between 1 and 100 : 68

Too low!

Enter a guess between 1 and 100 : 90

Too high!

Enter a guess between 1 and 100 : 80

Too high!

Enter a guess between 1 and 100 : 75

Correct! You got it in 6 guesses!

1. Write a program that prompts the user to enter a positive integer. Use a do-while loop to validate the input, ensuring the user re-enters the value until a valid positive integer is provided. Discuss why a do-while loop is more suitable for this scenario compared to a for or while loop.

**Code:-**

#include <iostream>

#include <cstdlib>

#include <string>

#include <fstream>

using namespace std;

int number;

int main()

{

do

{

cout << "Enter a positive number" << endl;

cin >> number;

}

while(number<0);

if (number > 0)

cout << "it is positive number " << endl;

return 0;

}

**Output:**-

Enter a positive number

-6

Enter a positive number

-24

Enter a positive number

45

it is positive number

1. You are developing a software module for a sports competition where scores are represented as integers. The competition has three events, and each participant can score between 0 to 100 points in each event. Your task is to write a C++ function to determine the highest score among the three events for a given participant, ensuring that your solution is efficient and readable.

**Code:-**

#include <iostream>

using namespace std;

int main() {

double event1, event2, event3;

for(int i=0;i<=2;i++)

{

cout << "\nEnter the scores of participant"<<i+1;

cout << "\n";

cin >> event1 >> event2 >> event3;

if(event1 >= event2 && event1 >= event3)

cout << "event1 has highest score: " << event1;

else if(event2 >= event1 && event2 >= event3)

cout << "event2 has highest score: " << event2;

else

cout << "event2 has highest score: " << event3;

}

return 0;

}

**Output:-**

Enter the scores of participant1

34

56

78

event2 has highest score: 78

Enter the scores of participant2

34

23

45

event2 has highest score: 45

Enter the scores of participant3

56

78

45

event2 has highest score: 78

4.Implement function overloading to handle three types of discounts:

A fixed amount discount off the total purchase price.

A percentage discount off the total purchase price.

A discount for bulk purchases: if the number of items is above a certain threshold, apply a fixed discount per item.

**Code:-**

#include <iostream>

#include <iomanip>

double calculateDiscountedPriceFixedAmount(double totalPrice, double fixedAmountDiscount) {

return totalPrice - fixedAmountDiscount;

}

double calculateDiscountedPricePercentage(double totalPrice, double percentageDiscount) {

return totalPrice - (totalPrice \* percentageDiscount / 100.0);

}

double calculateDiscountedPriceBulk(double totalPrice, double bulkDiscountPerItem, int itemCount, int itemCountThreshold) {

if (itemCount > itemCountThreshold) {

return totalPrice - (bulkDiscountPerItem \* (itemCount - itemCountThreshold));

} else {

return totalPrice;

}

}

int main() {

double originalPrice, fixedAmountDiscount, percentageDiscount, bulkDiscountPerItem;

int itemCount, itemCountThreshold;

std::cout << "Enter the original price: $";

std::cin >> originalPrice;

std::cout << "Enter the fixed amount discount: $";

std::cin >> fixedAmountDiscount;

std::cout << "Enter the percentage discount: %";

std::cin >> percentageDiscount;

std::cout << "Enter the bulk discount per item: $";

std::cin >> bulkDiscountPerItem;

std::cout << "Enter the item count: ";

std::cin >> itemCount;

std::cout << "Enter the item count threshold for bulk discount: ";

std::cin >> itemCountThreshold;

std::cout << std::fixed << std::setprecision(2);

std::cout << "Original Price: $" << originalPrice << std::endl;

std::cout << "Discounted Price with Fixed Amount Discount: $" << calculateDiscountedPriceFixedAmount(originalPrice, fixedAmountDiscount) << std::endl;

std::cout << "Discounted Price with Percentage Discount: $" << calculateDiscountedPricePercentage(originalPrice, percentageDiscount) << std::endl;

std::cout << "Discounted Price with Bulk Purchase Discount: $" << calculateDiscountedPriceBulk(originalPrice, bulkDiscountPerItem, itemCount, itemCountThreshold) << std::endl;

return 0;

}

**Output:-**

Enter the original price: $1000

Enter the fixed amount discount: $100

Enter the percentage discount: %20

Enter the bulk discount per item: $25

Enter the item count: 10

Enter the item count threshold for bulk discount: 10

Original Price: $1000.00

Discounted Price with Fixed Amount Discount: $900.00

Discounted Price with Percentage Discount: $800.00

Discounted Price with Bulk Purchase Discount: $1000.00

--------------------------------

Process exited after 32.29 seconds with return value 0

Press any key to continue . . .

5.Imagine you're developing a game where characters collect items that affect their stats and make decisions based on those stats. In this scenario, a character's health, experience, and inventory items will be influenced by various actions, demonstrating the use of different operators

**Code:-**

#include <iostream>

using namespace std;

int main() {

int health = 100;

int experience = 0;

int inventoryItems = 2;

experience += 50;

inventoryItems += 2;

int healthBoost = 0;

if (inventoryItems > 0) {

health += healthBoost;

inventoryItems--;

}

if (health <= 0) {

cout << "Game over! Your character has died." << endl;

} else if (experience >= 100) {

cout << "Congratulations! Your character leveled up!" << endl;

} else {

cout << "Keep playing! Your character is still alive and gaining experience." << endl;

}

cout << "Final status:" << endl;

cout << "Health: " << health << endl;

cout << "Experience: " << experience << endl;

cout << "Inventory items: " << inventoryItems << endl;

cout << "selected for the next round" << endl;

return 0;

}

**Output:-**

Keep playing! Your character is still alive and gaining experience.

Final status:

Health: 100

Experience: 50

Inventory items: 3

selected for the next round

--------------------------------

Process exited after 0.06647 seconds with return value 0

Press any key to continue . . .