Data Structures & Algorithms

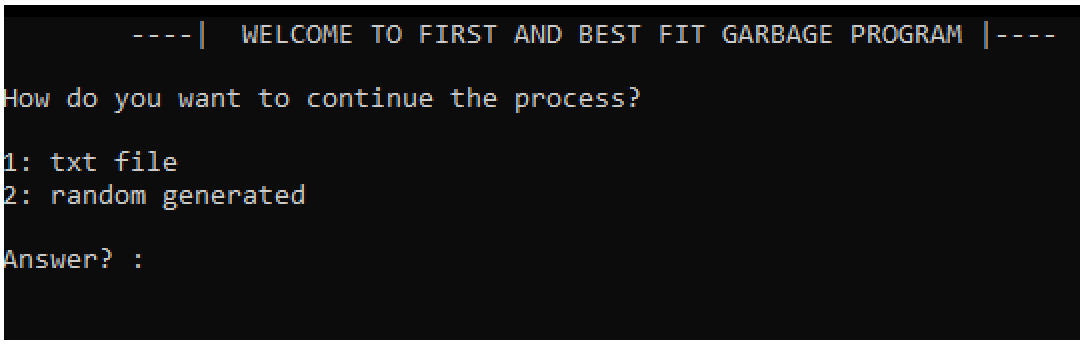
Term Project

Team String: 183217

Emrecan Üzüm	: 1900005485
Ahmet Kaan Memioğlu	: 1900005528

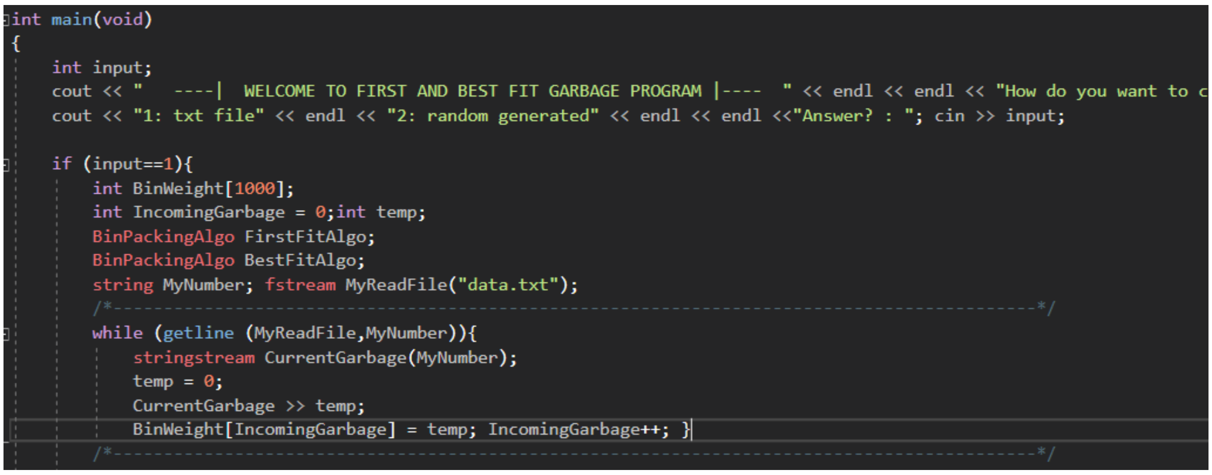
Our project is divided into three parts. We got 2 cpp files and a header file. In those files we have made a bin packing problem for best fit and first fit solutions.

In the **Main.cpp** file we are greeted with a user menu;



```
      ----|  WELCOME TO FIRST AND BEST FIT GARBAGE PROGRAM |----

How do you want to continue the process?

1: txt file
2: random generated

Answer? :
```

In our first condition we offer the user an input from the txt file which reads the bins weight and how many bins there are from the txt file alone.

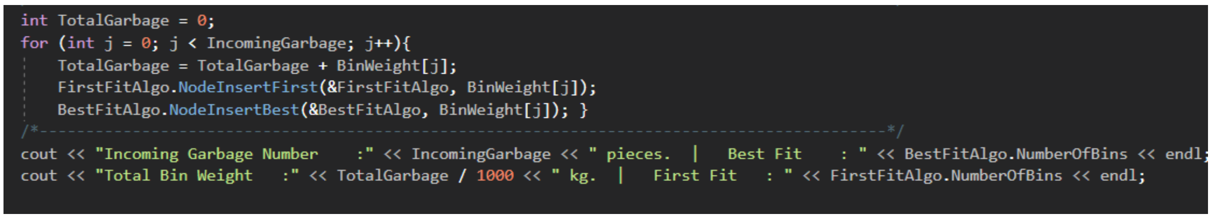


```
int main(void)
{
    int input;
    cout << "   ----|  WELCOME TO FIRST AND BEST FIT GARBAGE PROGRAM |----  " << endl << endl << "How do you want to c
    cout << "1: txt file" << endl << "2: random generated" << endl << endl <<"Answer? : "; cin >> input;

    if (input==1){
        int BinWeight[1000];
        int IncomingGarbage = 0;int temp;
        BinPackingAlgo FirstFitAlgo;
        BinPackingAlgo BestFitAlgo;
        string MyNumber; fstream MyReadFile("data.txt");
        /*-----------------------------------------------------------------------------------*/
        while (getline (MyReadFile,MyNumber)){
            stringstream CurrentGarbage(MyNumber);
            temp = 0;
            CurrentGarbage >> temp;
            BinWeight[IncomingGarbage] = temp; IncomingGarbage++; }
        /*-----------------------------------------------------------------------------------*/
```

Our BinWeight is an array which stores 1000 bins.After initialization of the array we are creating our algorithms objects FirstFit and BestFit. After that we are declaring MyNumber as a string variable and with the help of fstream function we are reading from the file “data.txt”.

Our loop is looping as long as it can read data from the txt file. With the help of “stringstream” we are changing the variable type by initializing a variable called “CurrenGarbage” to temp. After this process the index of our array is incremented by one and it’s index is changed into the weight of garbage.

After initializing the array successfully we are using the indexes of our array in a for loop. In this for loop we are transferring the arrays indexes (bins) to the algorithms meanwhile we are also calculating the total weight of garbage.

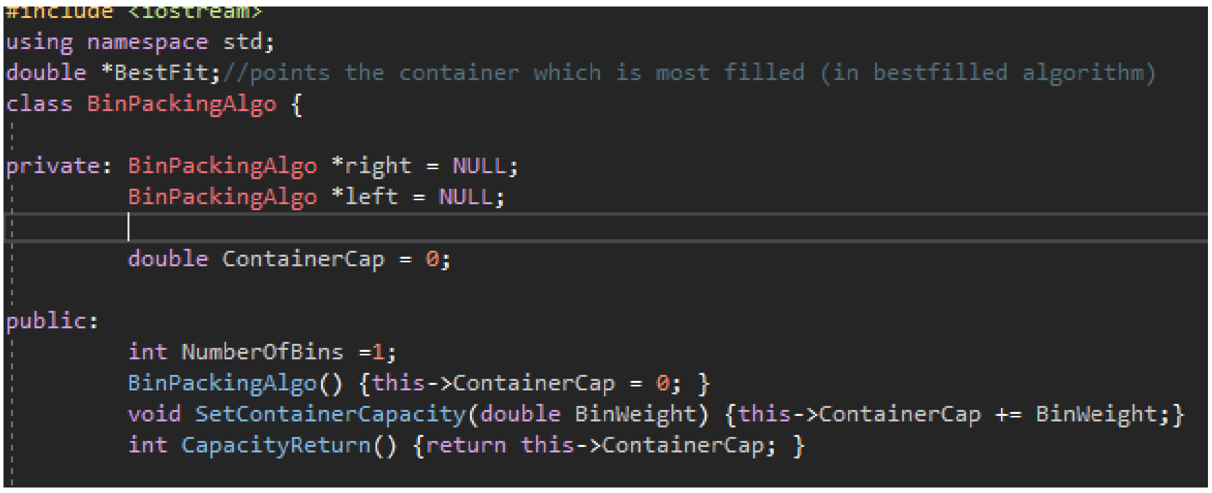


```
int TotalGarbage = 0;
for (int j = 0; j < IncomingGarbage; j++){
    TotalGarbage = TotalGarbage + BinWeight[j];
    FirstFitAlgo.NodeInsertFirst(&FirstFitAlgo, BinWeight[j]);
    BestFitAlgo.NodeInsertBest(&BestFitAlgo, BinWeight[j]); }
/*--------------------------------------------------------------------------------------------*/
cout << "Incoming Garbage Number    :" << IncomingGarbage << " pieces.  |   Best Fit    : " << BestFitAlgo.NumberOfBins << endl;
cout << "Total Bin Weight   :" << TotalGarbage / 1000 << " kg.  |   First Fit   : " << FirstFitAlgo.NumberOfBins << endl;
```

In our second condition starting off we are making the process random by srand(time(NULL)). We are initializing the IncomingGarbage's amount to a random number between 1000 and 1. In the remainder of that line we are also creating space (Memory Allocation) for the array of IncomingGarbage's weight which will be a random number between 1000 and 0. On the other hand the BinWeight's are added with TotalGarbage and its being equated to TotalGarbage with this way we can get the total weight of the Garbages .Our remaining for loop is pretty much the same with the for loop in condition one.

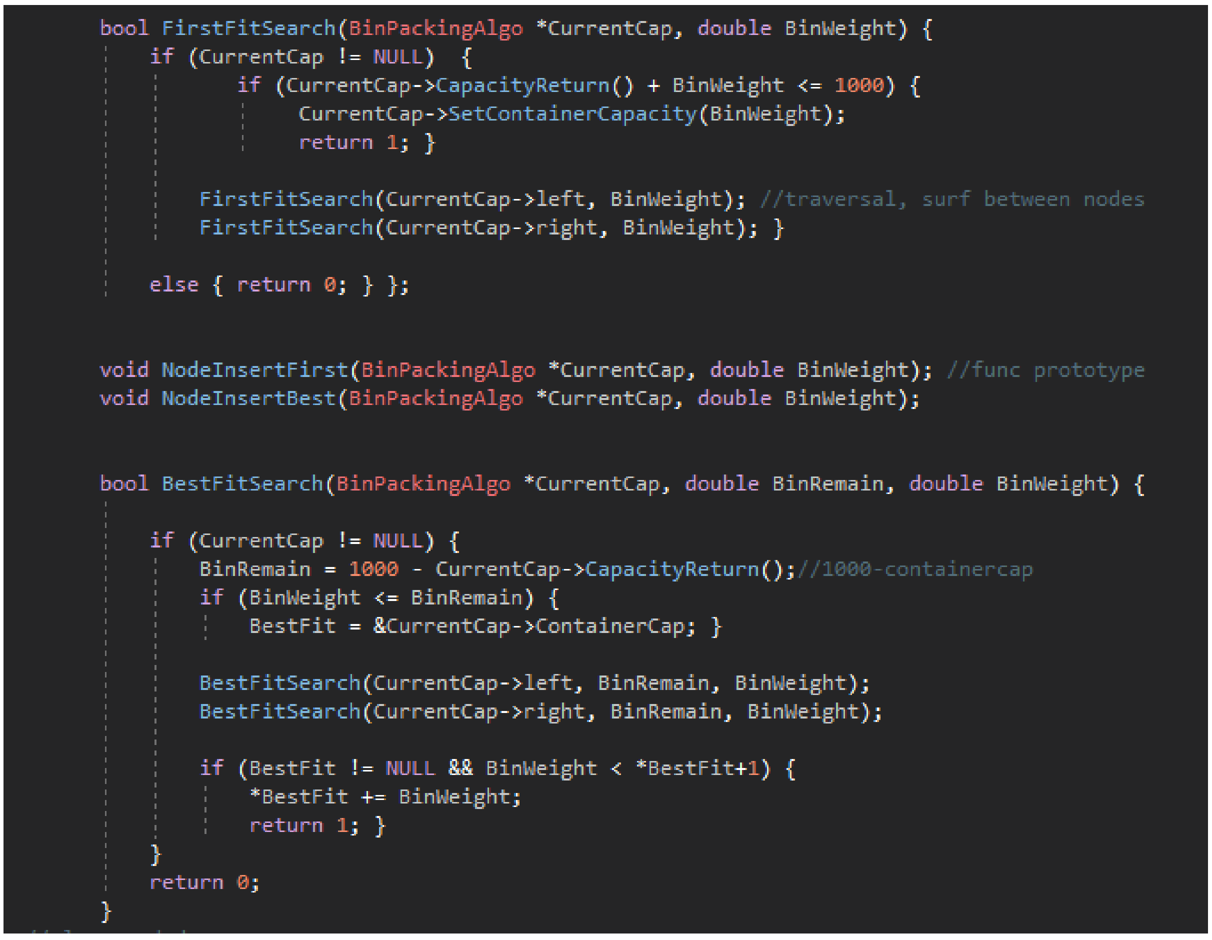
In our **BinPackingAlgo.cpp** file we are both creating algorithms and traversal functions for bin packing.

First things first we are creating a class and for traversal and inserting values we have to emulate a binary tree so we are creating nodes for our binary tree we both give them NULL values alongside with a double variable that holds the container capacity. Because we want no changes to our values in this bracket we are declaring these variables in the private section of the class. In our public section of the class we have declared a variable for bin number tracking called "NumberOfBins". We are also creating two functions that we are going to use for several lines after, called "SetContainerCapacity" and "CapacityReturn" these functions are being used for setting the capacity of containers and returning the capacity respectively.



```cpp
#include <iostream>
using namespace std;
double *BestFit;//points the container which is most filled (in bestfilled algorithm)
class BinPackingAlgo {

private: BinPackingAlgo *right = NULL;
         BinPackingAlgo *left = NULL;

         double ContainerCap = 0;

public:
         int NumberOfBins =1;
         BinPackingAlgo() {this->ContainerCap = 0; }
         void SetContainerCapacity(double BinWeight) {this->ContainerCap += BinWeight;}
         int CapacityReturn() {return this->ContainerCap; }
```

In our remainder of the class we are creating traversal functions and Insertion functions for the BestFit and FirstFit algorithms.

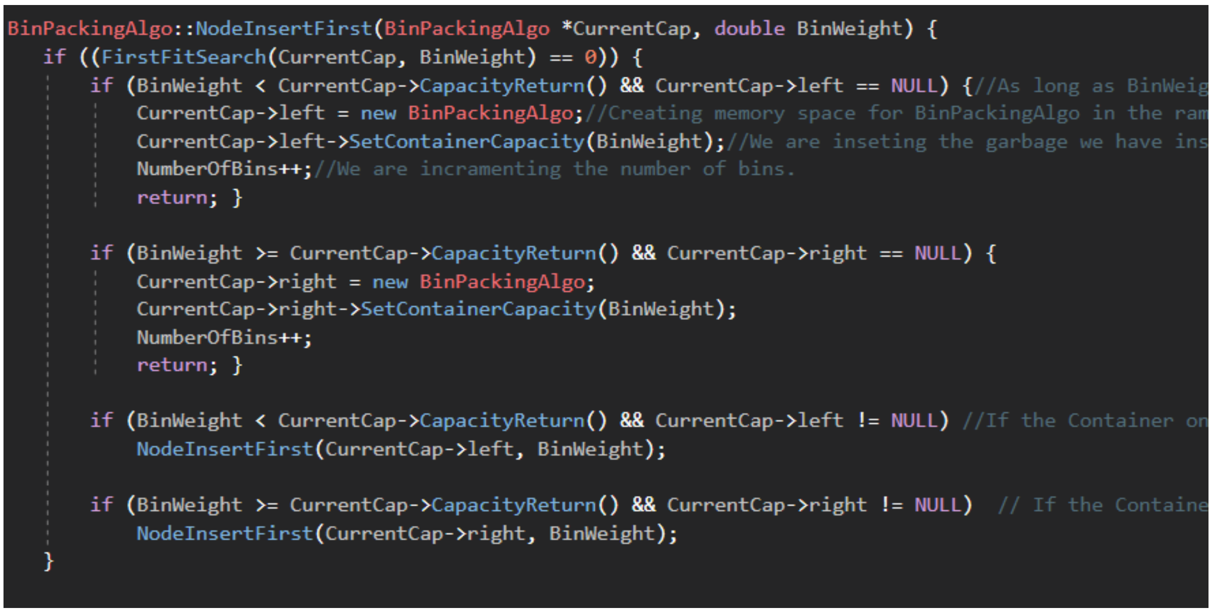


```cpp
bool FirstFitSearch(BinPackingAlgo *CurrentCap, double BinWeight) {
    if (CurrentCap != NULL)  {
        if (CurrentCap->CapacityReturn() + BinWeight <= 1000) {
            CurrentCap->SetContainerCapacity(BinWeight);
            return 1; }

        FirstFitSearch(CurrentCap->left, BinWeight); //traversal, surf between nodes
        FirstFitSearch(CurrentCap->right, BinWeight); }

    else { return 0; } };


void NodeInsertFirst(BinPackingAlgo *CurrentCap, double BinWeight); //func prototype
void NodeInsertBest(BinPackingAlgo *CurrentCap, double BinWeight);


bool BestFitSearch(BinPackingAlgo *CurrentCap, double BinRemain, double BinWeight) {

    if (CurrentCap != NULL) {
        BinRemain = 1000 - CurrentCap->CapacityReturn();//1000-containercap
        if (BinWeight <= BinRemain) {
            BestFit = &CurrentCap->ContainerCap; }

        BestFitSearch(CurrentCap->left, BinRemain, BinWeight);
        BestFitSearch(CurrentCap->right, BinRemain, BinWeight);

        if (BestFit != NULL && BinWeight < *BestFit+1) {
            *BestFit += BinWeight;
            return 1; }
    }
    return 0;
}
```

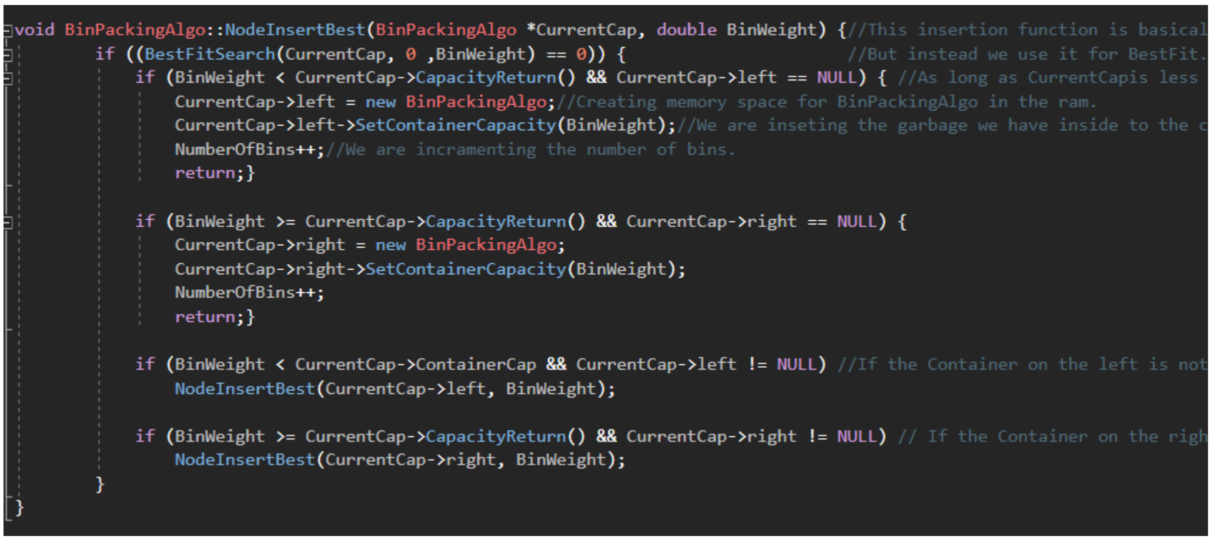
in the boolean function “FirstFitSearch” we are using loops inside of loops to traverse through the nodes and bins as long as CurrentCap is not NULL and as long as ContainerCap + BinWeight is less than or equal to 1000 set the incoming garbage’s weight to the binweight that we are currently using else we are returning zero which would we are not using this function because its a boolean function.

We are declaring the prototypes for Insertion functions so that we can use them in the future without having problems.

In our second boolean function "BestFitSearch" again we use the same method as long as CurrentCap is not NULL BinRemain variable(A variable that indicates space left for incoming garbage.) would be equal to 1000 (Max container weight) minus CurrentCap->CapacityReturn() which would mean 1000 - ContainerCap. We enter our first nested if statement which is as long as BinRemain is less than or and equal to BinWeight, BestFit for a garbage is calculated by &CurrentCap->ContainerCap. We traverse between the nodes in the next couple lines and after that the last if statement we have is the condition of whether BestFit is not equal to 0 which means there is space left and BestFits value +1 is less than BinWeight add the BinWeight to the BestFit for garbage and return 1 else return 0 and our class ends.

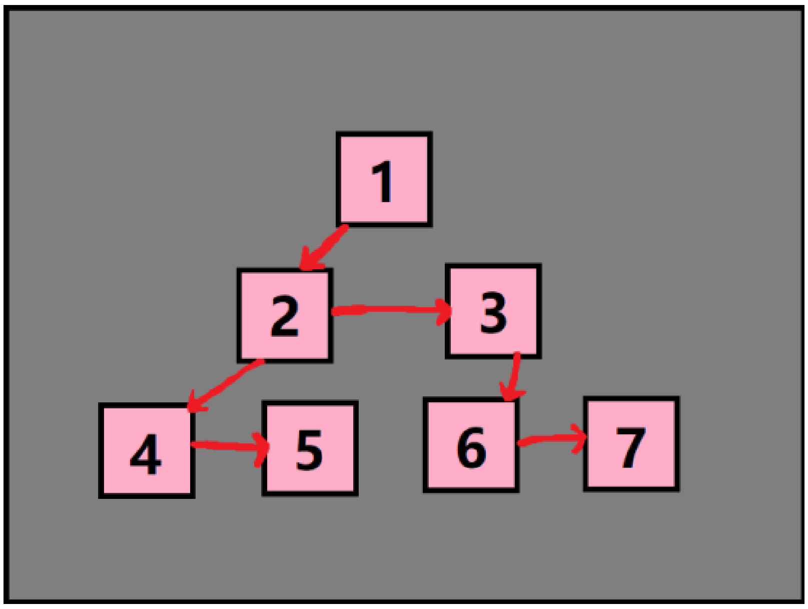


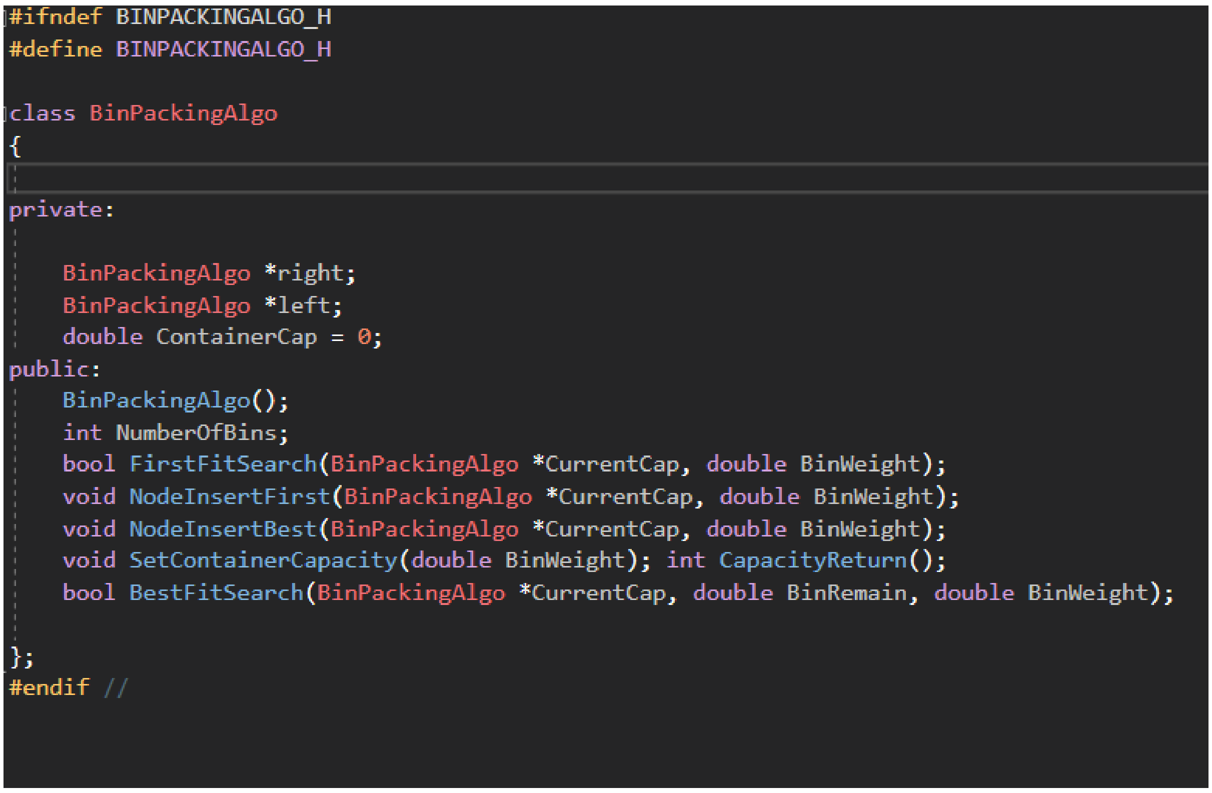
```
BinPackingAlgo::NodeInsertFirst(BinPackingAlgo *CurrentCap, double BinWeight) {
   if ((FirstFitSearch(CurrentCap, BinWeight) == 0)) {
       if (BinWeight < CurrentCap->CapacityReturn() && CurrentCap->left == NULL) {//As long as BinWeig
           CurrentCap->left = new BinPackingAlgo;//Creating memory space for BinPackingAlgo in the ram
           CurrentCap->left->SetContainerCapacity(BinWeight);//We are inseting the garbage we have ins
           NumberOfBins++;//We are incramenting the number of bins.
           return; }

       if (BinWeight >= CurrentCap->CapacityReturn() && CurrentCap->right == NULL) {
           CurrentCap->right = new BinPackingAlgo;
           CurrentCap->right->SetContainerCapacity(BinWeight);
           NumberOfBins++;
           return; }

       if (BinWeight < CurrentCap->CapacityReturn() && CurrentCap->left != NULL) //If the Container on
           NodeInsertFirst(CurrentCap->left, BinWeight);

       if (BinWeight >= CurrentCap->CapacityReturn() && CurrentCap->right != NULL)  // If the Containe
           NodeInsertFirst(CurrentCap->right, BinWeight);
   }
```



```cpp
void BinPackingAlgo::NodeInsertBest(BinPackingAlgo *CurrentCap, double BinWeight) {//This insertion function is basical
        if ((BestFitSearch(CurrentCap, 0 ,BinWeight) == 0)) {                          //But instead we use it for BestFit.
            if (BinWeight < CurrentCap->CapacityReturn() && CurrentCap->left == NULL) { //As long as CurrentCapis less
                CurrentCap->left = new BinPackingAlgo;//Creating memory space for BinPackingAlgo in the ram.
                CurrentCap->left->SetContainerCapacity(BinWeight);//We are inseting the garbage we have inside to the c
                NumberOfBins++;//We are incramenting the number of bins.
                return;}

            if (BinWeight >= CurrentCap->CapacityReturn() && CurrentCap->right == NULL) {
                CurrentCap->right = new BinPackingAlgo;
                CurrentCap->right->SetContainerCapacity(BinWeight);
                NumberOfBins++;
                return;}

            if (BinWeight < CurrentCap->ContainerCap && CurrentCap->left != NULL) //If the Container on the left is not
                NodeInsertBest(CurrentCap->left, BinWeight);

            if (BinWeight >= CurrentCap->CapacityReturn() && CurrentCap->right != NULL) // If the Container on the righ
                NodeInsertBest(CurrentCap->right, BinWeight);
        }
}
```

In our insertion functions we look for the nodes and we are going to place the weights of the garbage by looking though the left and right nodes. In our first if statement which is situation one as long as BinWeight is less than CurrentCapacity and if the left node is empty we are creating space for BinPackingAlgo in the ram and inserting our garbage we have inside to the container left and incrementing the number of bins. In our second if statement the process is exactly the same except we are looking through the right instead of left nodes.
In our third if statement we are inserting the weight value to the left node if the container on the left is not NULL and in our fourth and the last if statement we are doing the exact process but it is reversed so we look for the right nodes. With the insertion methods our binary tree will traverse through the nodes like this.

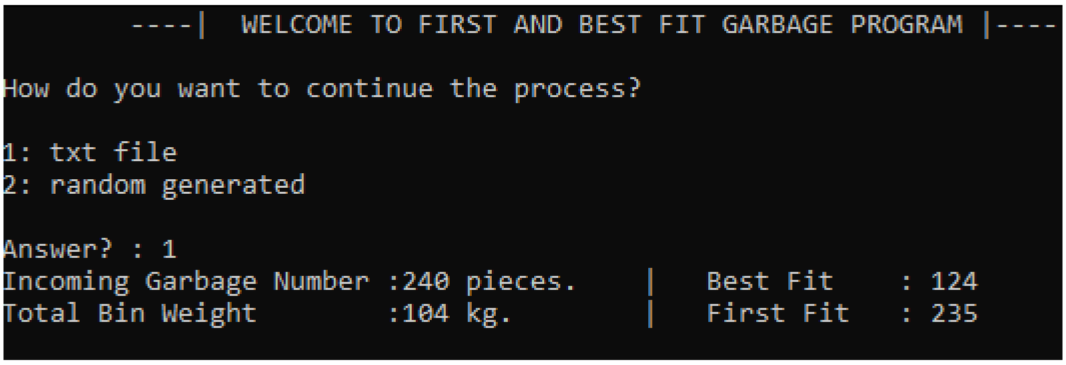




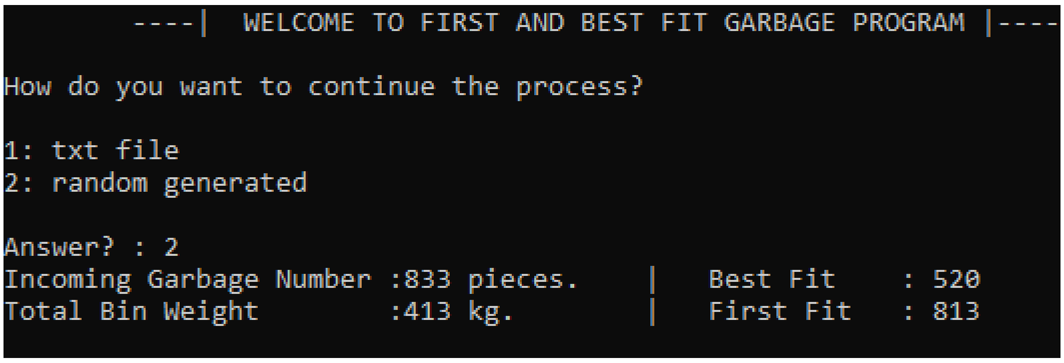
```cpp
#ifndef BINPACKINGALGO_H
#define BINPACKINGALGO_H

class BinPackingAlgo
{

private:

    BinPackingAlgo *right;
    BinPackingAlgo *left;
    double ContainerCap = 0;
public:
    BinPackingAlgo();
    int NumberOfBins;
    bool FirstFitSearch(BinPackingAlgo *CurrentCap, double BinWeight);
    void NodeInsertFirst(BinPackingAlgo *CurrentCap, double BinWeight);
    void NodeInsertBest(BinPackingAlgo *CurrentCap, double BinWeight);
    void SetContainerCapacity(double BinWeight); int CapacityReturn();
    bool BestFitSearch(BinPackingAlgo *CurrentCap, double BinRemain, double BinWeight);

};
#endif //
```

In our **BinPackingAlgo.h** file we have defined the values and the functions in BinPackingAlgo.cpp that we are going to use in Main.cpp.

OUTPUTS:



```
        ----|  WELCOME TO FIRST AND BEST FIT GARBAGE PROGRAM |----

How do you want to continue the process?

1: txt file
2: random generated

Answer? : 1
Incoming Garbage Number :240 pieces.     |   Best Fit    : 124
Total Bin Weight        :104 kg.         |   First Fit   : 235
```



```
        ----|  WELCOME TO FIRST AND BEST FIT GARBAGE PROGRAM |----

How do you want to continue the process?

1: txt file
2: random generated

Answer? : 2
Incoming Garbage Number :833 pieces.     |   Best Fit    : 520
Total Bin Weight        :413 kg.         |   First Fit   : 813
```