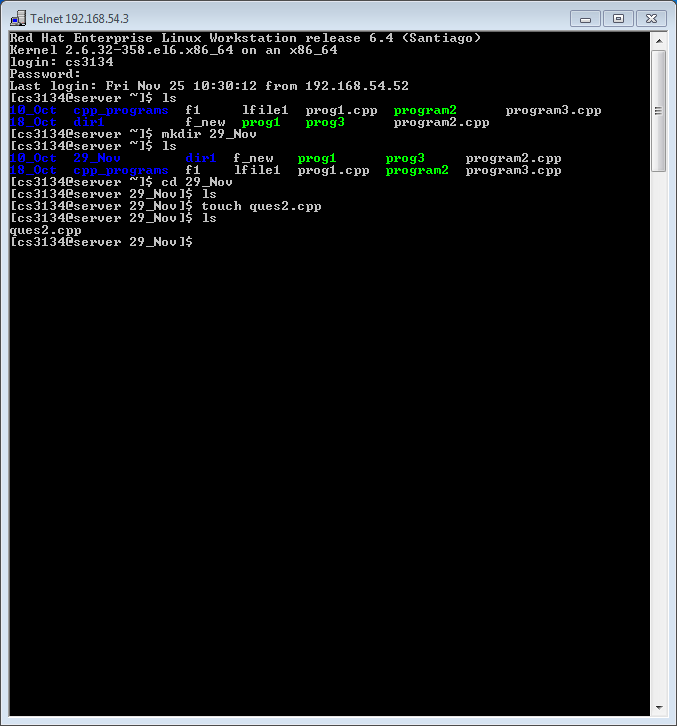
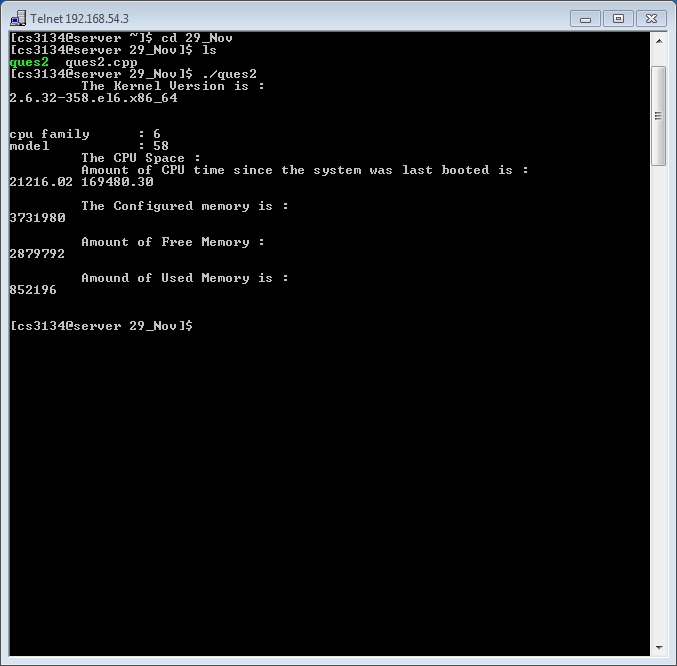
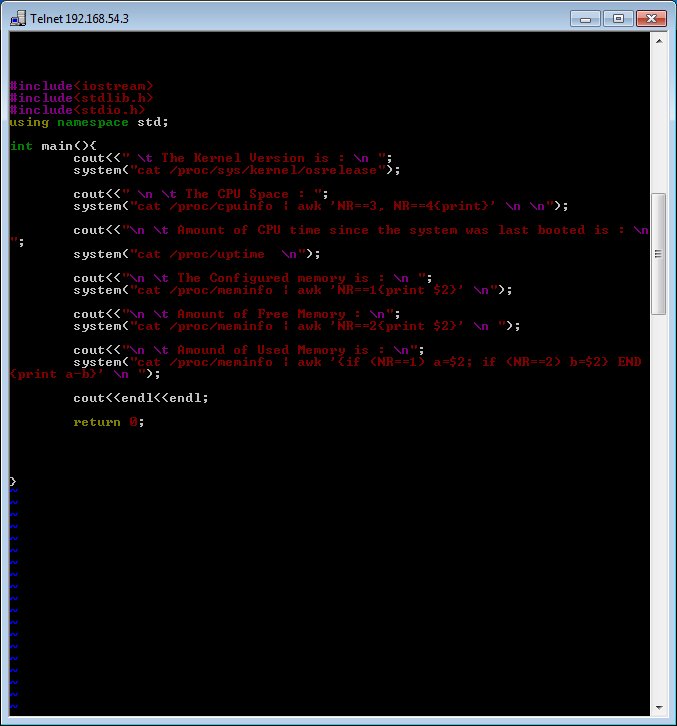
**PRACTICAL QUESTION 2 ::**

****

****

**Ques 13**

#include <iostream>

#include<bits/stdc++.h>

using namespace std;

void inputProcess(int processSize[], int processCount)

{

cout<<endl<<"-> Input the Process size below"<<endl;

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

{

cout<<"Enter size of Process "<<i+1<<" : ";

cin>>processSize[i];

}

cout<<endl;

return ;

}

void inputBlock(int blockSize[], int blockCount)

{

cout<<endl<<"-> Input the Memory Block size below"<<endl;

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

{

cout<<"Enter size of Memory Block "<<i+1<<" : ";

cin>>blockSize[i];

}

cout<<endl;

return ;

}

void displayBlock(int blockSize[], int blockCount)

{

cout<<endl<<"-> Displaying the Memory Block size below"<<endl;

cout<<"\t";

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

{

cout<<" | "<<blockSize[i];

}

cout<<endl;

return ;

}

void displayProcess(int processSize[], int processCount)

{

cout<<endl<<"-> Displaying the Process size below"<<endl;

cout<<"\t";

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

{

cout<<" | "<<processSize[i];

}

cout<<endl;

return ;

}

void worstFitAllocation(int blockSize[], int processSize[], int blockCount, int processCount)

{

int n=processCount;

int m=blockCount;

int allocation[n];

memset(allocation, -1, sizeof(allocation));

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

{

int wstIdx = -1;

for (int j=0; j<m; j++)

{

if (blockSize[j] >= processSize[i])

{

if (wstIdx == -1)

wstIdx = j;

else if (blockSize[wstIdx] < blockSize[j])

wstIdx = j;

}

}

// If we could find a block for current process

if (wstIdx != -1)

{

// allocate block j to p[i] process

allocation[i] = wstIdx;

// Reduce available memory in this block.

blockSize[wstIdx] -= processSize[i];

}

}

cout << "\nProcess No.\tProcess Size\tBlock no.\n";

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

{

cout << " " << i+1 << "\t\t" << processSize[i] << "\t\t";

if (allocation[i] != -1)

cout << allocation[i] + 1;

else

cout << "Not Allocated";

cout << endl;

}

}

void bestFitAllocation(int blockSize[], int processSize[], int blockCount, int processCount)

{

int n=processCount;

int m=blockCount;

int allocation[n];

// Initially no block is assigned to any process

memset(allocation, -1, sizeof(allocation));

// pick each process and find suitable blocks according to its size ad assign to it

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

{

// Find the best fit block for current process

int bestIdx = -1;

for (int j=0; j<m; j++)

{

if (blockSize[j] >= processSize[i])

{

if (bestIdx == -1)

bestIdx = j;

else if (blockSize[bestIdx] > blockSize[j])

bestIdx = j;

}

}

// If we could find a block for current process

if (bestIdx != -1)

{

// allocate block j to p[i] process

allocation[i] = bestIdx;

// Reduce available memory in this block.

blockSize[bestIdx] -= processSize[i];

}

}

cout << "\nProcess No.\tProcess Size\tBlock no.\n";

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

{

cout << " " << i+1 << "\t\t" << processSize[i] << "\t\t";

if (allocation[i] != -1)

cout << allocation[i] + 1;

else

cout << "Not Allocated";

cout << endl;

}

}

void firstFitAllocation(int blockSize[], int processSize[], int blockCount, int processCount)

{

int m=blockCount;

int n = processCount;

// block allocated to a process

int allocation[n];

// Initially no block is assigned to any process

memset(allocation, -1, sizeof(allocation));

// pick each process and find suitable blocks according to its size ad assign to it

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

{

for (int j = 0; j < m; j++)

{

if (blockSize[j] >= processSize[i])

{

// allocate block j to p[i] process

allocation[i] = j;

// Reduce available memory in this block.

blockSize[j] -= processSize[i];

break;

}

}

}

cout << "\nProcess No.\tProcess Size\tBlock no.\n";

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

{

cout << " " << i+1 << "\t\t"

<< processSize[i] << "\t\t";

if (allocation[i] != -1)

cout << allocation[i] + 1;

else

cout << "Not Allocated";

cout << endl;

}

}

void operationList()

{

cout<<endl<<"\t \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*"<<endl;

cout<<"\t MEMORY ALLOCATION ALGORITHMS AVAILABLE"<<endl;

cout<<"1. First Fit"<<endl;

cout<<"2. Best Fit"<<endl;

cout<<"3. Worst Fit"<<endl;

cout<<"\t \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*"<<endl;

return ;

}

int main()

{

cout << endl<<"\t \*\* PROGRAM TO IMPLEMENT MEMORY ALLOCATION ALGORITHMS \*\*" <<endl<< endl;

operationList();

char ch='y';

int choice;

int processSize[10];

int blockSize[10];

int processCount, blockCount;

while (ch=='y' or ch=='Y')

{

cout<<"-> Enter your choice (1/2/3) : ";

cin>>choice;

cout<<endl;

cout<<"-> Enter the number of processes : ";

cin>>processCount;

cout<<"-> Enter the number of memory blocks : ";

cin>>blockCount;

inputProcess(processSize, processCount);

inputBlock(blockSize, blockCount);

cout<<endl;

switch(choice)

{

case 1:

cout<<"\t \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* \n \t \t FIRST FIT"<<endl;

displayProcess(processSize, processCount);

displayBlock(blockSize, blockCount);

firstFitAllocation(blockSize, processSize, blockCount, processCount);

break;

case 2:

cout<<"\t \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* \n \t \t BEST FIT"<<endl;

displayProcess(processSize, processCount);

displayBlock(blockSize, blockCount);

bestFitAllocation(blockSize, processSize, blockCount, processCount );

break;

//int blockSize[], int processSize[], blockCount, int processCount

case 3:

cout<<"\t \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* \n \t \t WORST FIT"<<endl;

displayBlock(blockSize, blockCount);

worstFitAllocation(blockSize, processSize, blockCount, processCount);

break;

default:

cout<<"!! INVALID CHOICE !! TRY AGAIN !! "<<endl;

}

cout<<endl<<"Do you want to continue(y/n) : ";

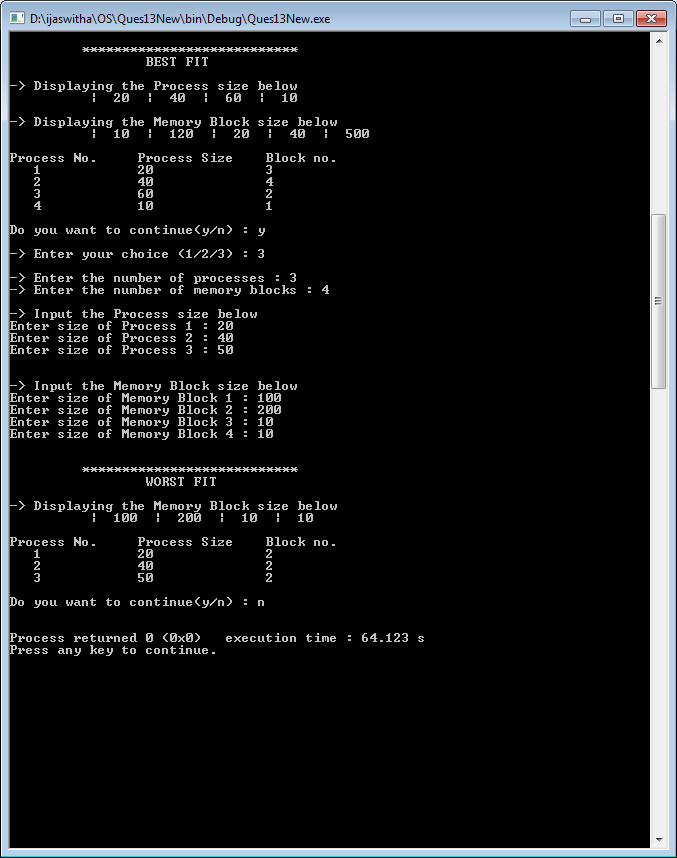
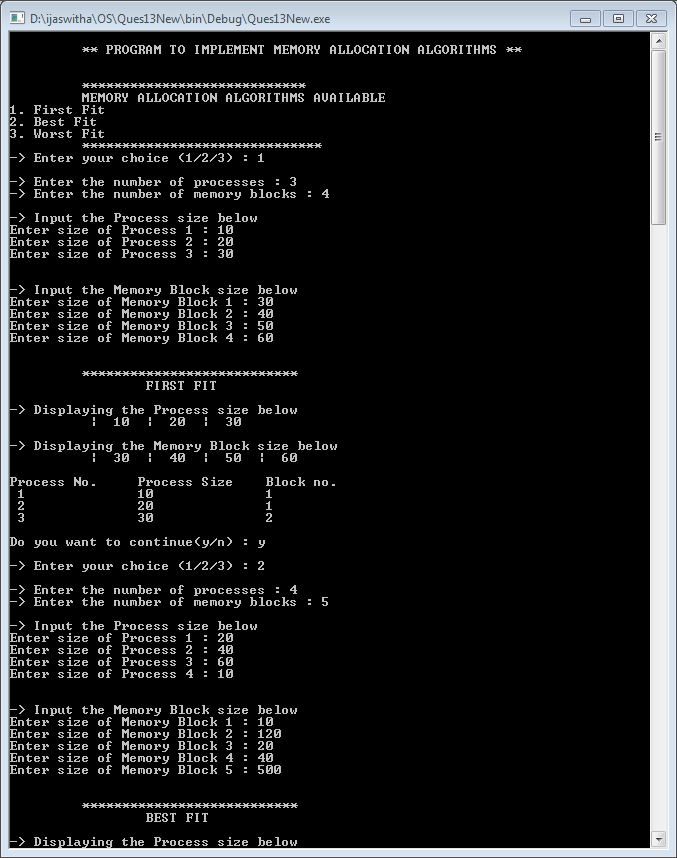
cin>>ch;

cout<<endl;

}

return 0;

}

s