Lab Experiments (Group Activity -1)

OPERATING SYSTEM

GROUP-7

Slot - B21 +B22+B23

Faculty – Dr.Abha Trivedi

GROUP MEMBERS:

- 20BCE10077 - FIZA SIDDIQUI
- 20BCE10091 - AARUSHI JAIN
- 20BCE10093 - ANANYA PRASAD
- 20BCE10102 - SUGANDHA KUMARI
- 20BCE10232 - RAAZ MAURYA
- 20BCE10256 - AYUSH BHATT

A) Contiguous Allocation Techniques Implementation of Contiguous allocation techniques:

- **Worst-Fit**
- **Best-Fit**
- **First-Fit**

B) External and Internal Fragmentation Calculation of external and internal fragmentation.

1. CODE FOR WORST FIT:

```
#include <iostream>
using namespace std;

int main()
{
   cout<<"\nWorst Fit"<<endl;
   int n_blocks, n_files;

   cout << "\nNumber of free blocks: ";
   cin >> n_blocks;

   cout << "Number of files: ";
   cin >> n_files;

   int used=0;
   int total=0;

   int file[n_files], block[n_blocks];
```

```
  bool filled[n_blocks];
  for (int i = 0; i < n_blocks; i++)
  {
     filled[i] = false;
  }

  int isAllocated[n_files];
  for (int i = 0; i < n_files; i++)
  {
     isAllocated[i] = -1;
  }

  int fragment[n_files];
  for (int i = 0; i < n_files; i++)
  {
     fragment[i] = -1;
  }

  cout << "\nEnter size of: "<<endl;
  for (int i = 0; i < n_blocks; i++)
  {
     cout << "Block " << i + 1 << ": ";
     cin >> block[i];
     total+=block[i];
  }

  cout << "\nEnter size of: "<<endl;
  for (int i = 0; i < n_files; i++)
  {
     cout << "File " << i + 1 << ": ";
     cin >> file[i];
```

```
    }

    int internalFrag = 0;

    for (int i = 0; i < n_files; i++)
    {
        int temp=INT32_MIN;
        int tempAllocation = -1;
        for (int j = 0; j < n_blocks; j++)
        {
            if (file[i] < block[j] && filled[j] == false)
            {
                if(temp<(block[j]-file[i]))
                {
                    temp = block[j]-file[i];
                    tempAllocation = j;
                }
            }
        }
        if(tempAllocation!=-1)
            {
                filled[tempAllocation]=true;
                isAllocated[i]=tempAllocation;
                fragment[i] = temp;
                internalFrag+=temp;
                used+=file[i];
            }
    }
    int unallocatedFile=0;
    cout << "\nFile No.\tFile Size\tBlock No.\tBlock Size\tInternal Fragmentation" << endl;
    for (int i = 0; i < n_files; i++)
```

```
  {
    cout << i + 1 << "\t\t";
    cout << file[i] << "\t\t";
    if (isAllocated[i] != -1)
    {
      cout << isAllocated[i] + 1 << "\t\t";
      cout << block[isAllocated[i]] << "\t\t";
      cout << fragment[i] << endl;
    }
    else
    {
      unallocatedFile += file[i];
      cout << "Unallocated"
        << "\t";
      cout << "-\t\t";
      cout << "-" << endl;
}
}

  int remaining = total - used;

  cout<<"\nTotal Internal Fragmentation: "<<internalFrag<<endl;

  if (unallocatedFile > 0 && remaining >= unallocatedFile)
  {
    cout << "External Fragmentation:" << endl;
    for (int i = 0; i < n_files; i++)
    {
      if (isAllocated[i] == -1)
      {
```

```
            cout << "File " << i+1 << " of size " << file[i] << " cannot be allocated to fragmented space of size " << remaining << endl;
        }
      }
   }
   else if(unallocatedFile==0)
   {
      cout<<"\nAll files are allocated."<<endl;
   }
   cout<<endl;


   return 0;
}
```

OUTPUT FOR WORST FIT:

```
Worst Fit

Number of free blocks: 5
Number of files: 4

Enter size of:
Block 1: 600
Block 2: 500
Block 3: 200
Block 4: 300
Block 5: 100

Enter size of:
File 1: 110
File 2: 420
File 3: 215
File 4: 425

File No.        File Size       Block No.       Block Size      Internal Fragmentation
1               110             1               600             490
2               420             2               500             80
3               215             4               300             85
4               425             Unallocated     -               -

Total Internal Fragmentation: 655
External Fragmentation:
File 4 of size 425 cannot be allocated to fragmented space of size 955
```

2. CODE FOR BEST FIT:

```
#include <iostream>
using namespace std;

int main()
{
   cout<<"\nBest Fit"<<endl;
   int n_blocks, n_files;

   cout << "\nNumber of free blocks: ";
   cin >> n_blocks;

   cout << "Number of files: ";
   cin >> n_files;

   int used=0;
   int total=0;


   int file[n_files], block[n_blocks];

   bool filled[n_blocks];
   for (int i = 0; i < n_blocks; i++)
   {
      filled[i] = false;
   }

   int isAllocated[n_files];
   for (int i = 0; i < n_files; i++)
   {
      isAllocated[i] = -1;
   }

   int fragment[n_files];
   for (int i = 0; i < n_files; i++)
   {
      fragment[i] = -1;
   }

   cout << "\nEnter size of: "<<endl;
   for (int i = 0; i < n_blocks; i++)
   {
      cout << "Block " << i + 1 << ": ";
      cin >> block[i];
      total+=block[i];
```

```
    }

    cout << "\nEnter size of: "<<endl;
    for (int i = 0; i < n_files; i++)
    {
       cout << "File " << i + 1 << ": ";
       cin >> file[i];
    }

    int internalFrag = 0;

    for (int i = 0; i < n_files; i++)
    {
       int temp=INT32_MAX;
       int tempAllocation = -1;
       for (int j = 0; j < n_blocks; j++)
       {
          if (file[i] < block[j] && filled[j] == false)
          {
             if(temp>(block[j]-file[i]))
             {
                temp = block[j]-file[i];
                tempAllocation = j;
             }
          }
       }
       if(tempAllocation!=-1)
          {
             filled[tempAllocation]=true;
             isAllocated[i]=tempAllocation;
             fragment[i] = temp;
             internalFrag+=temp;
             used+=file[i];
          }
    }
    int unallocatedFile=0;
    cout << "\nFile No.\tFile Size\tBlock No.\tBlock Size\tInternal Fragmentation"
<< endl;
    for (int i = 0; i < n_files; i++)
    {
       cout << i + 1 << "\t\t";
       cout << file[i] << "\t\t";
       if (isAllocated[i] != -1)
       {
          cout << isAllocated[i] + 1 << "\t\t";
          cout << block[isAllocated[i]] << "\t\t";
```

```
            cout << fragment[i] << endl;
        }
        else
        {
            unallocatedFile += file[i];
            cout << "Unallocated"
                 << "\t";
            cout << "-\t\t";
            cout << "-" << endl;
        }
    }

    int remaining = total - used;

    cout<<"\nTotal Internal Fragmentation: "<<internalFrag<<endl;

    if (unallocatedFile > 0 && remaining >= unallocatedFile)
    {
        cout << "External Fragmentation:" << endl;
        for (int i = 0; i < n_files; i++)
        {
            if (isAllocated[i] == -1)
            {
                cout << "File " << i+1 << " of size " << file[i] << " cannot be allocated
to fragmented space of size " << remaining << endl;
            }
        }
    }
    else if(unallocatedFile==0)
    {
        cout<<"\nAll files are allocated."<<endl;
    }
    cout<<endl;

    return 0;
}
```

OUTPUT FOR BEST FIT:

```
Best Fit

Number of free blocks: 5
Number of files: 4

Enter size of:
Block 1: 600
Block 2: 500
Block 3: 200
Block 4: 300
Block 5: 100

Enter size of:
File 1: 110
File 2: 420
File 3: 215
File 4: 425

File No.        File Size       Block No.       Block Size      Internal Fragmentation
1               110             3               200             90
2               420             2               500             80
3               215             4               300             85
4               425             1               600             175

Total Internal Fragmentation: 430

All files are allocated.
```

3. *CODE FOR FIRST FIT:*

```
#include <iostream>
using namespace std;

int main()
{
   cout << "\nFirst Fit" << endl;

   int n_blocks, n_files;

   cout << "\nNumber of free blocks: ";
   cin >> n_blocks;

   cout << "Number of files: ";
   cin >> n_files;

   int used = 0;
   int total = 0;

   int file[n_files], block[n_blocks];

   bool filled[n_blocks];
   for (int i = 0; i < n_blocks; i++)
   {
      filled[i] = false;
   }

   int isAllocated[n_files];
   for (int i = 0; i < n_files; i++)
   {
      isAllocated[i] = -1;
   }

   int fragment[n_files];
   for (int i = 0; i < n_files; i++)
   {
      fragment[i] = -1;
   }

   cout << "\nEnter size of: " << endl;
   for (int i = 0; i < n_blocks; i++)
   {
      cout << "Block " << i + 1 << ": ";
      cin >> block[i];
      total += block[i];
```

```
    }

    cout << "\nEnter size of: " << endl;
    for (int i = 0; i < n_files; i++)
    {
       cout << "File " << i + 1 << ": ";
       cin >> file[i];
    }

    int internalFrag = 0;

    for (int i = 0; i < n_files; i++)
    {
       for (int j = 0; j < n_blocks; j++)
       {
          if (file[i] < block[j] && filled[j] == false)
          {
             filled[j] = true;
             isAllocated[i] = j;
             fragment[i] = block[j] - file[i];
             internalFrag += block[j] - file[i];
             used += file[i];
             break;
          }
       }
    }
    int unallocatedFile = 0;
    cout << "\nFile No.\tFile Size\tBlock No.\tBlock Size\tInternal Fragmentation" <<
endl;
    for (int i = 0; i < n_files; i++)
    {
       cout << i + 1 << "\t\t";
       cout << file[i] << "\t\t";
       if (isAllocated[i] != -1)
       {
          cout << isAllocated[i] + 1 << "\t\t";
          cout << block[isAllocated[i]] << "\t\t";
          cout << fragment[i] << endl;
       }
       else
       {
          unallocatedFile += file[i];
          cout << "Unallocated"
               << "\t";
          cout << "-\t\t";
          cout << "-" << endl;
```

```
        }
    }

    int remaining = total - used;

    cout << "\nTotal Internal Fragmentation: " << internalFrag << endl;

    if (unallocatedFile > 0 && remaining >= unallocatedFile)
    {
        cout << "External Fragmentation:" << endl;
        for (int i = 0; i < n_files; i++)
        {
            if (isAllocated[i] == -1)
            {
                cout << "File " << i + 1 << " of size " << file[i] << " cannot be allocated to
fragmented space of size " << remaining << endl;
            }
        }
    }
    else if (unallocatedFile == 0)
    {
        cout << "\nAll files are allocated." << endl;
    }
    cout << endl;

    return 0;
}
```

OUTPUT FOR FIRST FIT:

```
First Fit

Number of free blocks: 5
Number of files: 4

Enter size of:
Block 1: 600
Block 2: 500
Block 3: 200
Block 4: 300
Block 5: 100

Enter size of:
File 1: 110
File 2: 420
File 3: 215
File 4: 425

File No.        File Size       Block No.       Block Size      Internal Fragmentation
1               110             1               600             490
2               420             2               500             80
3               215             4               300             85
4               425             Unallocated     -               -

Total Internal Fragmentation: 655
External Fragmentation:
File 4 of size 425 cannot be allocated to fragmented space of size 955
```