OS Phase 1
Module 2 - GD, PD, H instruction implementation

Name: Bhavin Patil
Roll No.: 66

```c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

int readLine(FILE *fptr, char *buffer)
{
   // ^ is an indication for the inverted set for the given scanset
   int res = fscanf(fptr, "%[^\n]", buffer);
   getc(fptr);
   return res;
}

void init(char *M, int size)
{
   memset(M, '\0', size * sizeof(char));
}

void writeLine(FILE *fptr, char *content)
{
   fprintf(fptr, "%s", content);
   fputc('\n', fptr);
}

void executeUserProgram(FILE *fReadPtr, int *IC, char *IR, char *R, int
*C, char (*M)[4], char *buffer)
{
   {
       int SI = false;
       FILE *fWritePtr = fopen("output.txt", "w");
```

```
      while (true)
      {
          memcpy(IR, M[*IC], 4);

          if (IR[0] == 'G' && IR[1] == 'D')
          {
              SI = 1;
              int res = readLine(fReadPtr, buffer);
              int start = ((IR[2] - 48) * 10) + IR[3] - 48;
              memcpy(M[start], buffer, 10 * 4);
          }
          else if (IR[0] == 'P' && IR[1] == 'D')
          {
              SI = 2;
              int start = (IR[2] - 48) * 10 + IR[3] - 48;
              char blockContent[40];
              memcpy(blockContent, M[(int)start], 10 * 4);
              memset(IR, '\0', 4);
              writeLine(fWritePtr, blockContent);
          }
          else if (IR[0] == 'H')
          {
              SI = 3;
              putc('\n', fWritePtr);
              putc('\n', fWritePtr);
              break;
          }
          *IC = *IC + 1;
      }
   }
}
void startExecution(FILE *fReadPtr, char *IR, char *R, int *C, char *M,
char *buffer)
{
   int IC = 0;
   executeUserProgram(fReadPtr, &IC, IR, R, C, *M, buffer);
}

int main()
```

```
{
   char M[100][4];
   char R[4];
   char IR[4];
   int IC;
   int C;
   char buffer[41];

   FILE *fptr = fopen("./input.txt", "r");

   while (!feof(fptr))
   {
       int res = readLine(fptr, buffer);

       if ((buffer[0] == '$') && (buffer[1] == 'A'))
       {
           init(*M, 100 * 4);
           init(R, 4);
           init(IR, 4);

           res = readLine(fptr, buffer);

           int i = 0;
           int offset = 0;
           char instruction[4];

           while (1)
           {
               char first = buffer[offset];
               if (first == 'H')
               {
                   M[i++][0] = buffer[offset];
                   break;
               }
               else
               {
                   memcpy(M[i++], buffer + offset, 4);
                   offset += 4;
               }
           }
```

```
        }
        else if ((buffer[0] == '$') && (buffer[0] == 'D'))
        {
            startExecution(fptr, IR, R, &C, *M, buffer);
        }
        else if ((buffer[0] == '$') && (buffer[0] == 'E'))
        {
        }
    }

    fclose(fptr);
    return 0;
}
```