Grupo: 2
Carlos Henrique F. L. da Fonseca
Hádamo da Silva Egito
Wellerson Prenholato de Jesus
Geraldo Nogueira

Trabalho de Estrutura de Dados I
Relatório

1. Compilação:

Suponhamos que o usuário esteja utilizando o Sistema Operacional Linux.
Primeiramente iremos acessar no terminal a pasta onde encontra-se todos arquivos que foram encaminhados através do email. Arquivos esses nomeados com: main.c, permutar.c, permutar.h, fila.c, fila.h, pilha.c, pilha.h e Makefile.
Sucessivamente utilizaremos o Makefile para fazer a compilação do trabalho.
Obs.: O objetivo de Makefile é definir regras de compilação para projetos de software.
Depois de acessarmos a pasta com os arquivos através do terminal, iremos digitar "make" na linha de comando do terminal,logo ele irá gerar um executável com nome de exec na pasta selecionada anteriormente. Desse modo podemos executá-lo na linha de comando, ainda dentro da pasta digitamos "./exec".
Obs.:O programa make interpreta o conteúdo do Makefile e executa as regras lá definidas.
Após ter digitado "./exec" temos dois métodos de entrada de dados no programa.
No primeiro método, podemos digitar a entrada na linha de comando no terminal como exemplo "./exec preta abc verde def amarela gh", sendo assim "preta" a cor da primeira caixa e "abc" as respectivas bolas contidas nessa caixa, sucessivamente "verde" seria a cor da próxima caixa e "def" as bolas contidas nessa caixa, logo "amarela" seria a última caixa "gh" as bolas contidas na caixa.
No segundo método, caso não seja digitada a entrada na linha de comando do terminal ou por falta de informação, iremos fazer a inserção manual de acordo com o que é solicitado pelo sistema.

2. Estruturas:

2.1 - Estrutura usada na função recursiva de permutação:

```
typedef struct permutacao{
     fila** caixa; // Armazena a(s) caixa(s)
     pilha** perm; // Pilha que contém a permutação
     int qnt_caixa; // Quantidade de caixa(s)
} permutacao; // Estrutura que armazena vetores de pilha e fila
```

2.2 - Tipo de dados abstrato Fila:

```
typedef struct fila{
     char* cor; // Cor da caixa
     int quantidade; // Quantidade de bola(s) na Fila (Caixa)
     bolaF* ini; // Ponteiro para a primeira bola da Fila
     bolaF* fim; // Ponteiro para a ultima bola da Fila
} fila; // TAD Fila

typedef struct bolaF {
     char elem; // Armazena o char que representa a letra
     struct bolaF* prox; // Ponteiro para próxima bolaF
     struct bolaF* ant; // Ponteiro para bolaF anterior
} bolaF; // Estrutura bolaF armazenada da Fila
```

2.3 - Tipo de dados abstrato Pilha:

```
typedef struct pilha{
     char* cor; // Cor da caixa
     int quantidade; // Quantidade de bola(s) na Pilha (Caixa)
     bolaP* ini; // Ponteiro para a ultima bola da Pilha
     bolaP* fim; // Ponteiro para a ultima bola da Pilha
} pilha; // TAD Pilha

typedef struct bolaP {
     char elem;// Armazena o char que representa a letra
     struct bolaP* prox; // Ponteiro para próxima bolaP
     struct bolaP* ant; // Ponteiro para bolaP anterior
} bolaP; // Estrutura bolaP armazenada da Pilha
```

As estruturas “bolaP” e “bolaF” representam as bolas que sao armazenadas respectivamente nas estruturas “Pilha” e “Fila”, onde a “Pilha” representa a permutação e a “Fila” a caixa. A estrutura “permutacao” vetores para as “Fila” e “Pilha”, e o tamanho do vetor.

3. Funções

3.1 – char charEntrada();

Função que verifica se a entrada é um char. Usa fgets e verifica pelo tamanho da string e pela função isalpha da <ctype.h>, para confirmar se é uma letra e retorna o char, caso não seja, pede para inserir novamente. É usada na main para entrada das letras das bolas.

3.2 - int inteiroEntrada();

Faz verificação da entrada, se é realmente um número inteiro. Usa fgets para entrada de uma string e faz conversão com a função strtol, usando o ponteiro para char e verifica se o resto da conversão feita pela strtol é diferente de '\n', caso seja diferente o valor informado é inválido e pede para inserir novamente. É usada na main para entrada de valores.

3.3 - void stringEntrada(char*);

Função de entrada de uma string com de correção de erros de digitação, como apenas um espaço ou apenas 'Enter'. É usada na main para pedir a cor da caixa.

3.4 - void Ordena(fila,int);**

Função de ordenação das fila em ordem não decrescente usando o Bubble Sort. É chamada na main após a inserção dos dados do programa.

3.5 - void permutar(fila, int);**

Função que recebe o vetor que contém a(s) caixa(s) e aloca a estrutura 'permutacao' e a(s) pilha(s) necessária(s) na permutar_rec. É chamada na main.

3.6 - void permutar_rec(permutacao*, int);

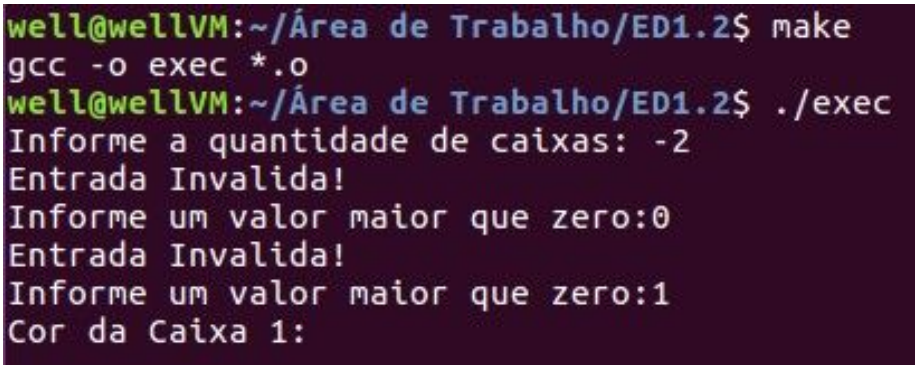
Função que faz as permutações de todas as caixas recursivamente. Usando o métodos da fila e pilha para inserção e remoção e chamadas recursivas para a mesma caixa gera uma permutação, assim, verifica se há uma próxima caixa para gerar uma permutação, se houver, chama recursivamente aumentando o contador um 1 para a próxima caixa, caso não houver, imprime as permutações de cada caixa naquele momento. É chamada apenas pela função permutar.

3.7 - Funções básicas de gerenciamento de Pilha e Fila.

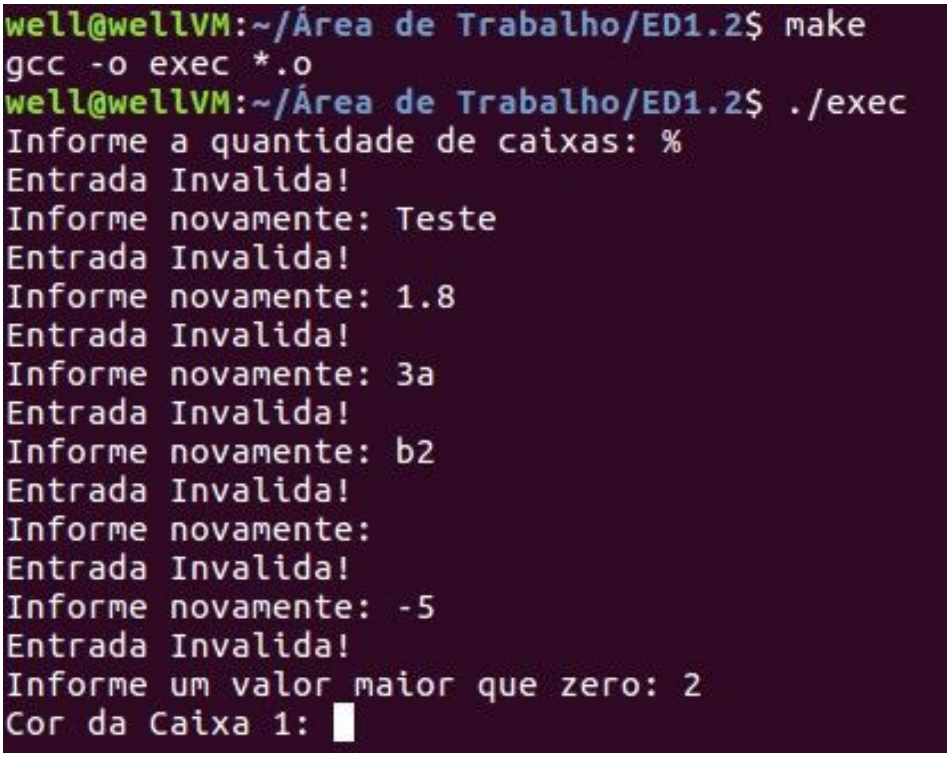
4. Testes

Principais casos de testes identificados:

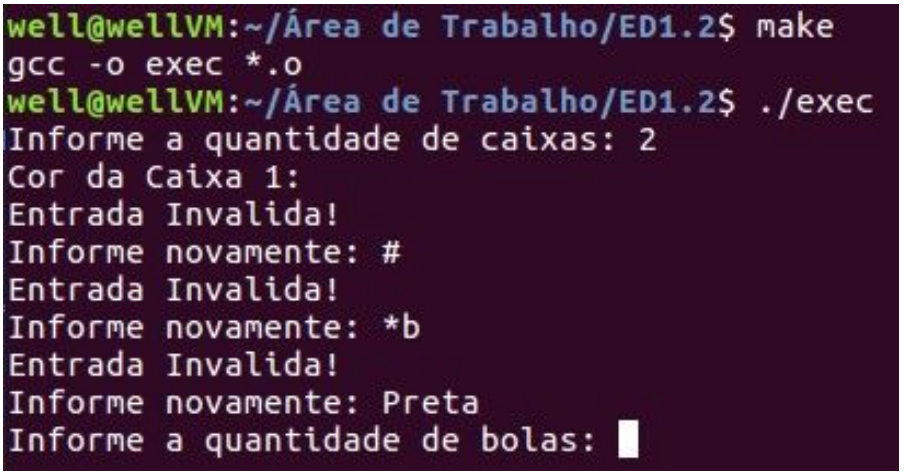
4.1 - Caso o número de caixas seja menor ou igual a zero, então o usuário receberá uma mensagem para inserir um novo valor válido. Esse valor será válido caso seja maior que zero.



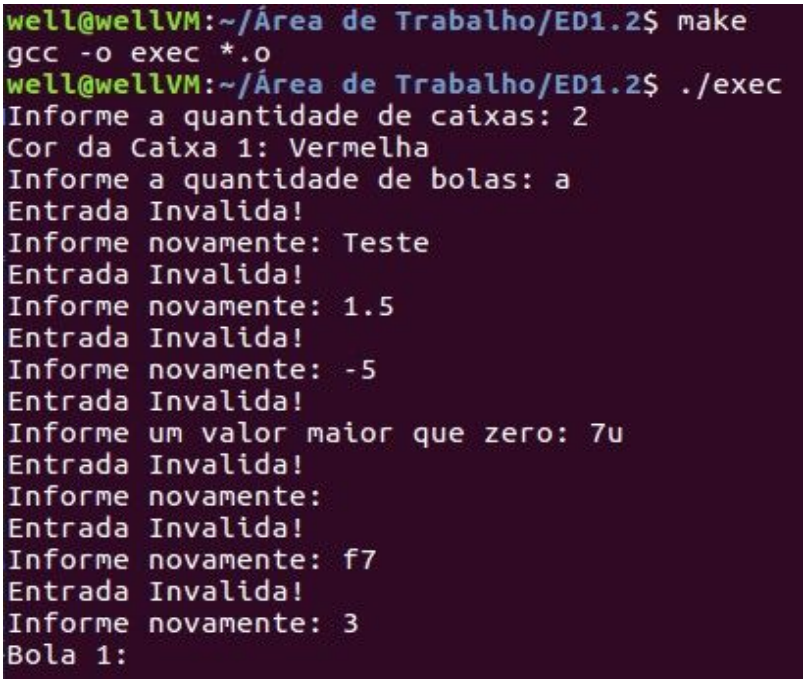
4.2 - Caso no momento que o sistema solicitar o número de caixas e o usuário inserir um caractere, string, número decimal, número negativo, letra acompanhada de um número, número acompanhado de uma letra ou vazio, então o usuário receberá uma mensagem para inserir um novo valor válido. Esse valor será válido apenas para números inteiros maiores que zero.



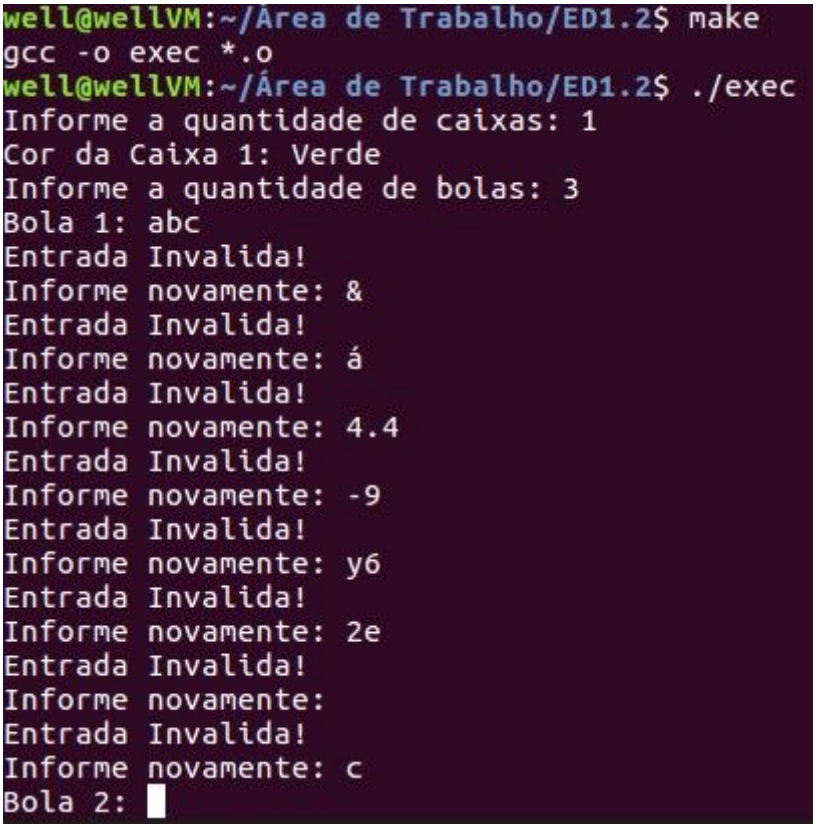
4.3 - Caso no momento que o sistema solicitar a cor de cada caixa e o usuário inserir enter ou começar a string com algum caractere especial, então o usuário receberá uma mensagem para inserir um novo valor válido. Esse valor será válido desde que o usuário não comece a string com caracteres especiais.



4.4 - Caso no momento que o sistema solicitar a quantidade de bolas de cada caixa e o usuário digitar um caractere, string, número decimal, número negativo, letra acompanhada de um número, número acompanhado de uma letra ou vazio, então o usuário receberá uma mensagem para inserir um novo valor válido. Esse valor será válido apenas para números inteiros maiores que zero.



4.5 - Caso no momento que o sistema solicitar as letras de cada bola referentes a uma determinada caixa e o usuário digitar uma string, caractere especial(letras com acento incluídas), número decimal, número negativo, letra acompanhada de um número, número acompanhado de uma letra ou vazio, então o usuário receberá uma mensagem para inserir um novo valor válido. Esse valor será válido apenas para uma única letra, desde que essa letra seja do alfabeto.



4.6 - Outros possíveis casos testados:

1 caixa 1 bola.
1 caixa 3 bolas.
2 caixas 1 bola cada.
2 caixas 3 bolas cada.
2 caixas uma com 3 bolas outra com 2 bolas.
5 caixas com quantidades diferentes de bolas.

Nos casos citados acima, o programa funciona normalmente.

Obs.: E para conclusão do trabalho, vários outros testes foram executados.